Continual Machine Learning
Summer 2024

Teacher

Dr. Martin Mundt,

Research Group on Open World Lifelong Learning

Time

Every Friday 14:25 - 16:05 CEST

Course Homepage

http://owll-lab.com/teaching/cl_lecture_24

https://www.youtube.com/playlist?list=PLm6QXeaB-XkA5-lVBB-h7XeYzFzgSh6sk

OWLL ContinualAI hessian.AI TECHNISCHE UNIVERSITÄT DARMSTADT

Week 2: Transfer Learning, Domain Adaptation & Continual/Lifelong Machine Learning

Early definition: lifelong ML

Definition - **Lifelong Machine Learning** - Thrun 1996:

"The system has performed N tasks. When faced with the (N+1)th task, it uses the knowledge gained from the N tasks to help the (N+1)th task."

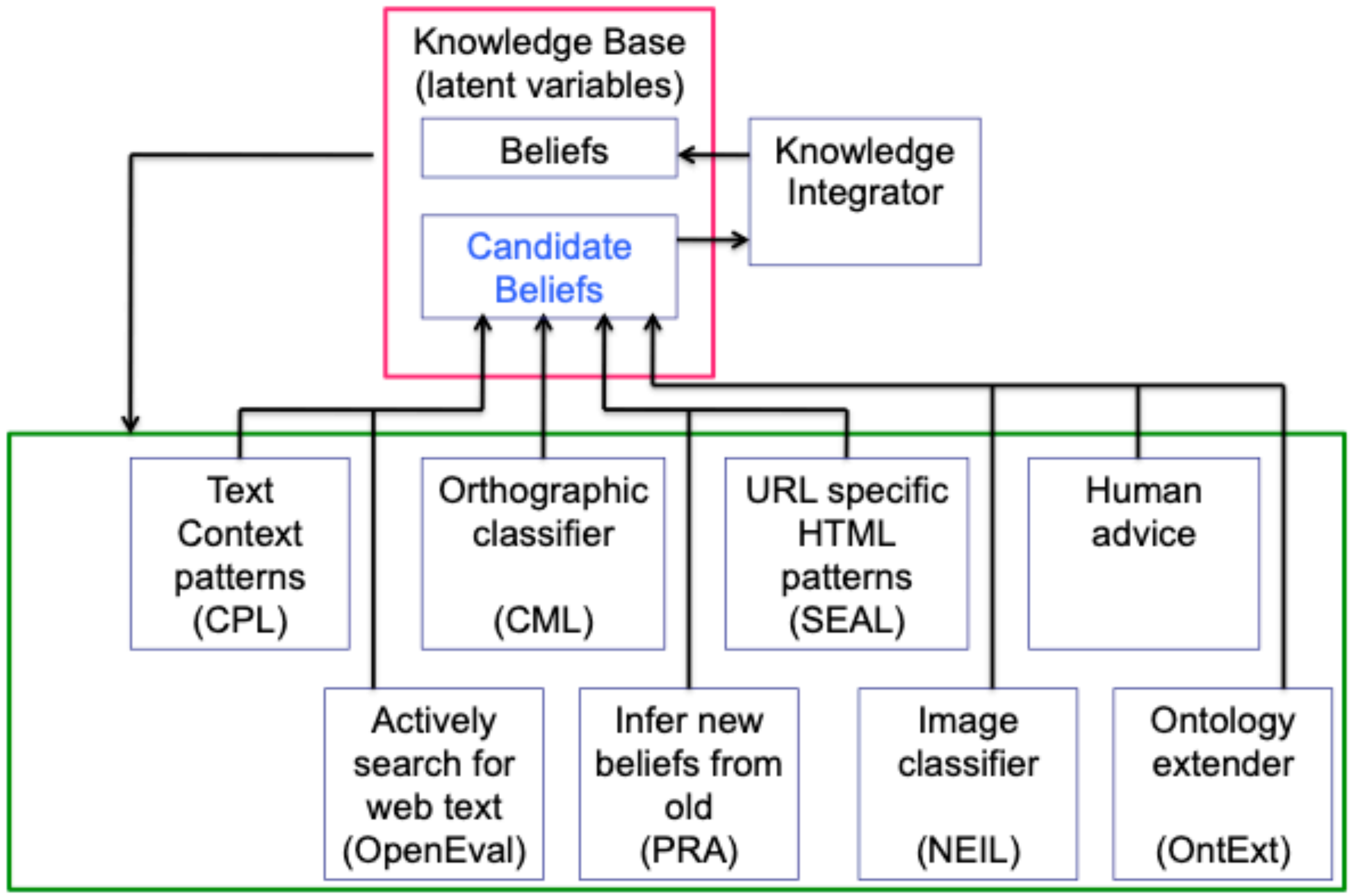
"Is Learning The n-th Thing Any Easier Than Learning the First?" (NeurIPS 1996) & "Explanation based Neural Network Learning A Lifelong Learning Approach", Springer US, 1996

What is *knowledge* in a machine learning system?

Never-ending language learner

Knowledge is a lot more than just parameters

NELL Architecture



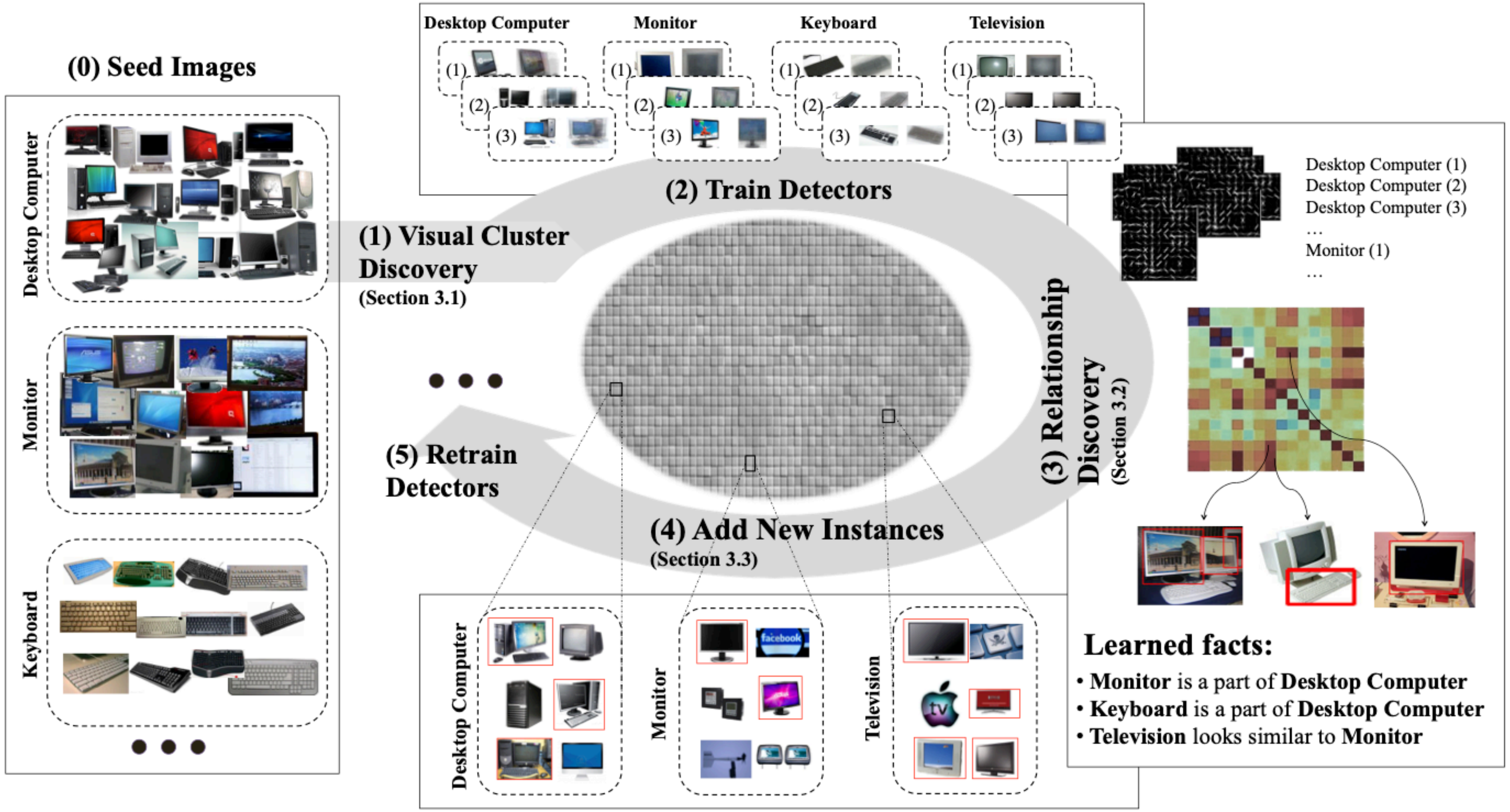
- Ran 24/7 from 2010-2018
- Accumulated over 50 million candidate "beliefs" by reading the web
- Relational database
- Facts: barley is a grain
- Beliefs: sportUsesEquip (soccer, balls)

"Towards an Architecture for Never-Ending Language Learning", Carlson et al, AAAI 2010

"Never-Ending Learning", T. Mitchell et al, AAAI 2015

Never-ending image learner

Knowledge is a lot more than just parameters



"NEIL: Extracting Visual Knowledge form Web Data", X. Chen et al, ICCV 2013

Early definition: lifelong ML

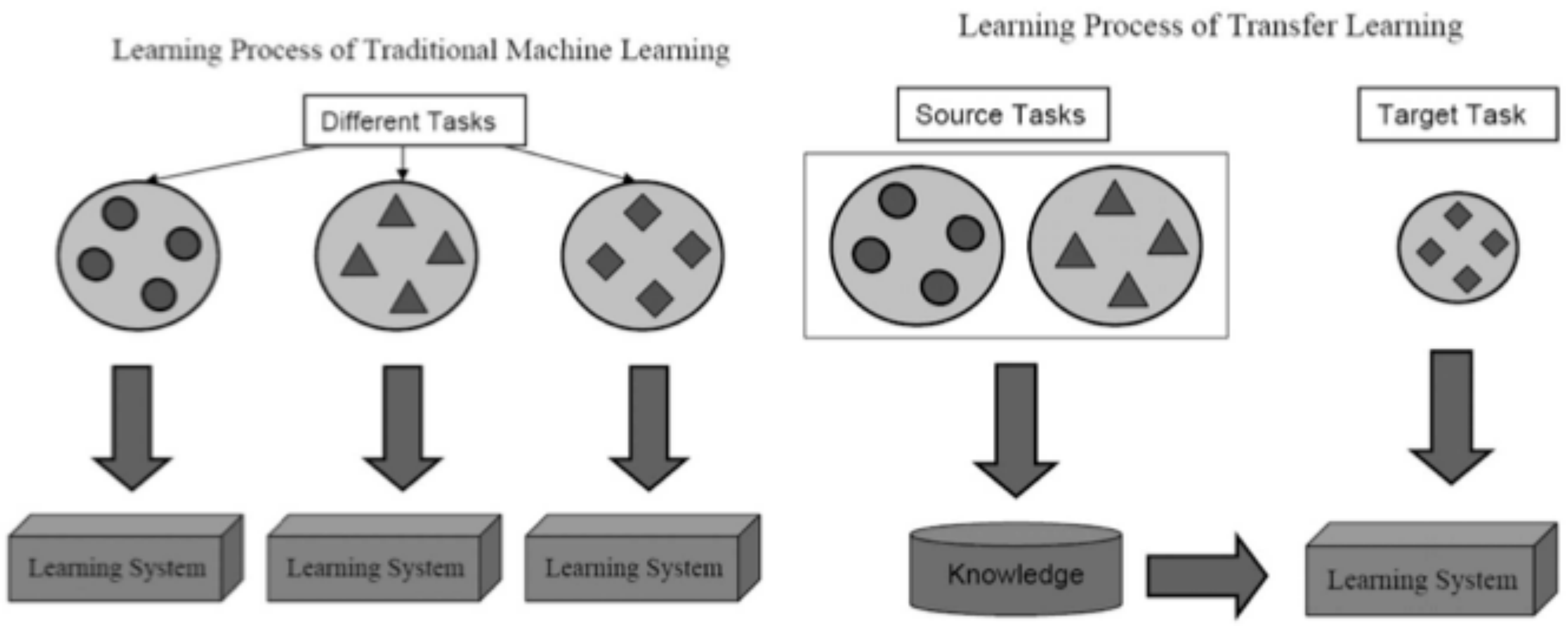
Definition - **Lifelong Machine Learning** - Thrun 1996:

“The system has performed N tasks. When faced with the (N+1)th task, it uses the knowledge gained from the N tasks to help the (N+1)th task.”

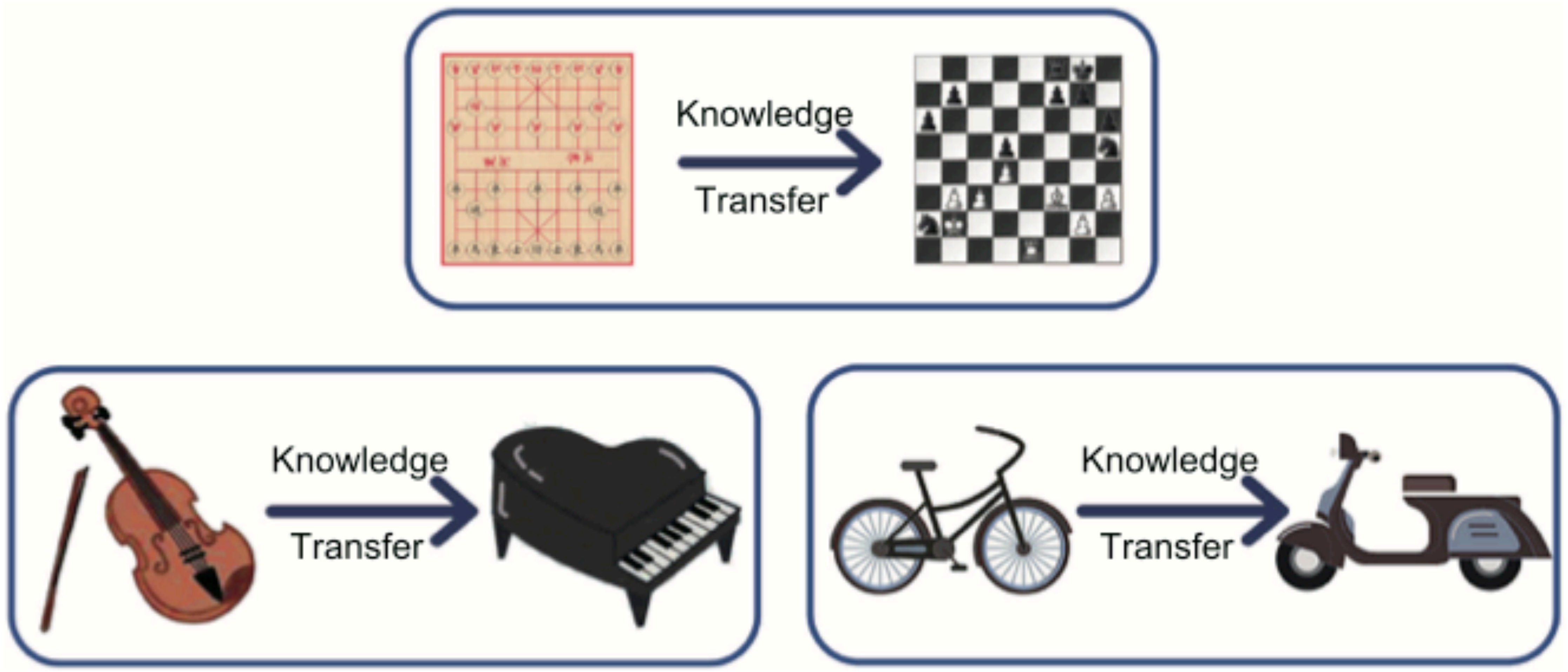
- Is data accumulated? Stored?
- What are the ways to “help” the (N+1)th task?
- What is knowledge? What is a task?
- ….

“Is Learning The n-th Thing Any Easier Than Learning the First?” (NeurIPS 1996) & “Explanation based Neural Network Learning A Lifelong Learning Approach”, Springer US, 1996

Transfer learning



"A Survey on Transfer Learning", Pan and Yang, IEEE Transactions on Knowledge & Data Engineering, 2010

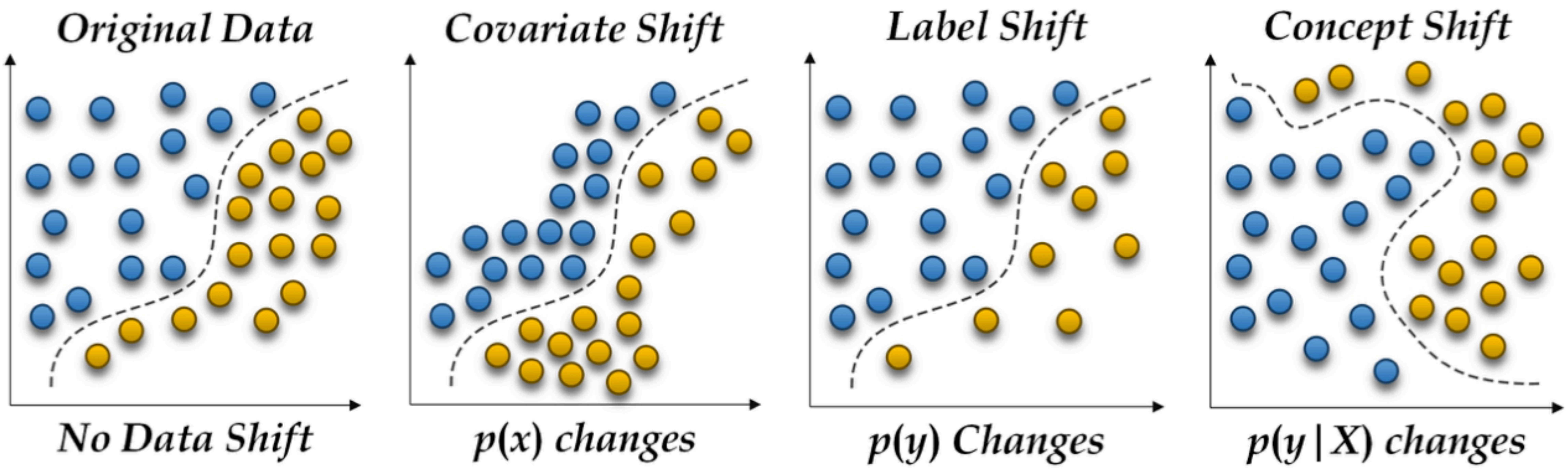


"A Comprehensive Survey on Transfer Learning", Zhuang et al, Proceedings of IEEE, 2020

"Help the (N+1th) task!": Assume that we already have "knowledge"/ a model based on initial task(s) -> the essence of **transfer learning**

What types of *shifts* can you think of?

Dataset shifts



(a) Original data (b) Covariate shift (c) Label shift (d) Concept shift

Figure from “Understanding Dataset Shift and Potential Remedies”, Vector Institute Technical Report, 2021

See also: “Dataset Shift in Machine Learning” book, MIT Press 2009

Transfer learning: definition

Definition - Transfer Learning - Pan & Yang 2009:
"*Given a source domain* D_S *and learning task* $\mathcal{T}_S$, *a target domain* D_T *and learning task* $\mathcal{T}_T$, *transfer learning aims to help improve the learning of the target predictive function* $f_T(\,.\,)$ *in* D_T *using the knowledge in* D_S *and* $\mathcal{T}_S$, *where* $D_S \neq D_T$ *or* $\mathcal{T}_s \neq \mathcal{T}_T$."

- Domain D
- Task $\mathcal{T}$
- Source S
- Target T

"A Survey on Transfer Learning", Pan & Yang, IEEE Transactions on Knowledge and Data Engineering 22(10), 2009

Transfer learning: definition

Definition - Domain & Task - Pan & Yang 2009:

"Given a specific domain, $D = \{\mathcal{X}, p(x)\}$*, a task consists of two components: a label space Y and an objective predictive function* $f()$ *(denoted by* $T = \{Y, f()\}$*, which is not observed but can be learned from the training data, which consist of pairs* $\{x^{(n)}, y^{(n)}\}$*, where* $x^{(n)} \in X$ and $y^{(n)} \in Y$*."*

- Domain D: a pair of data distribution $p(x)$ and corresponding feature space $\mathcal{X}$
- Task $\mathcal{T}$: find a function f() (to map to labels in the case of supervision)
- Where generally $\mathcal{X}_S \neq \mathcal{X}_T$ or $p_S(x) \neq p_T(x)$

"A Survey on Transfer Learning", Pan & Yang, IEEE Transactions on Knowledge and Data Engineering 22(10), 2009

Transductive transfer

Definition - Transductive Transfer Learning - Pan & Yang 2009:
"*Given a source domain* D_S *and learning task* $\mathcal{T}_S$, *a target domain* D_T *and learning task* $\mathcal{T}_T$, *transductive transfer learning aims to help improve the learning of the target predictive function* $f_T(\,.\,)$ *in* D_T *using the knowledge in* D_S *and* $\mathcal{T}_S$, *where* $D_S \neq D_T$ *and* $\mathcal{T}_s = \mathcal{T}_T$."

- Feature spaces between the source and target are different $\mathcal{X}_S \neq \mathcal{X}_T$
- Feature spaces between source and target are the same, but $p_S(x) \neq p_T(x)$
- Frequently encountered as **domain adaptation** or **sample selection bias**

"A Survey on Transfer Learning", Pan & Yang, IEEE Transactions on Knowledge and Data Engineering 22(10), 2009

Inductive transfer

Definition - Inductive Transfer Learning - Pan & Yang 2009:
"*Given a source domain* D_S *and learning task* $\mathcal{T}_S$, *a target domain* D_T *and learning task* $\mathcal{T}_T$, *inductive transfer learning aims to help improve the learning of the target predictive function* $f_T(\,.\,)$ *in* D_T *using the knowledge in* D_S *and* $\mathcal{T}_S$, *where* $\mathcal{T}_s \neq \mathcal{T}_T$."

(A few) (labeled) data points are required to "induce" the target predictive function

"A Survey on Transfer Learning", Pan & Yang, IEEE Transactions on Knowledge and Data Engineering 22(10), 2009

What do you think are the central questions & measures of success for transfer learning?

Transfer: questions & goals

(Some) **central questions**

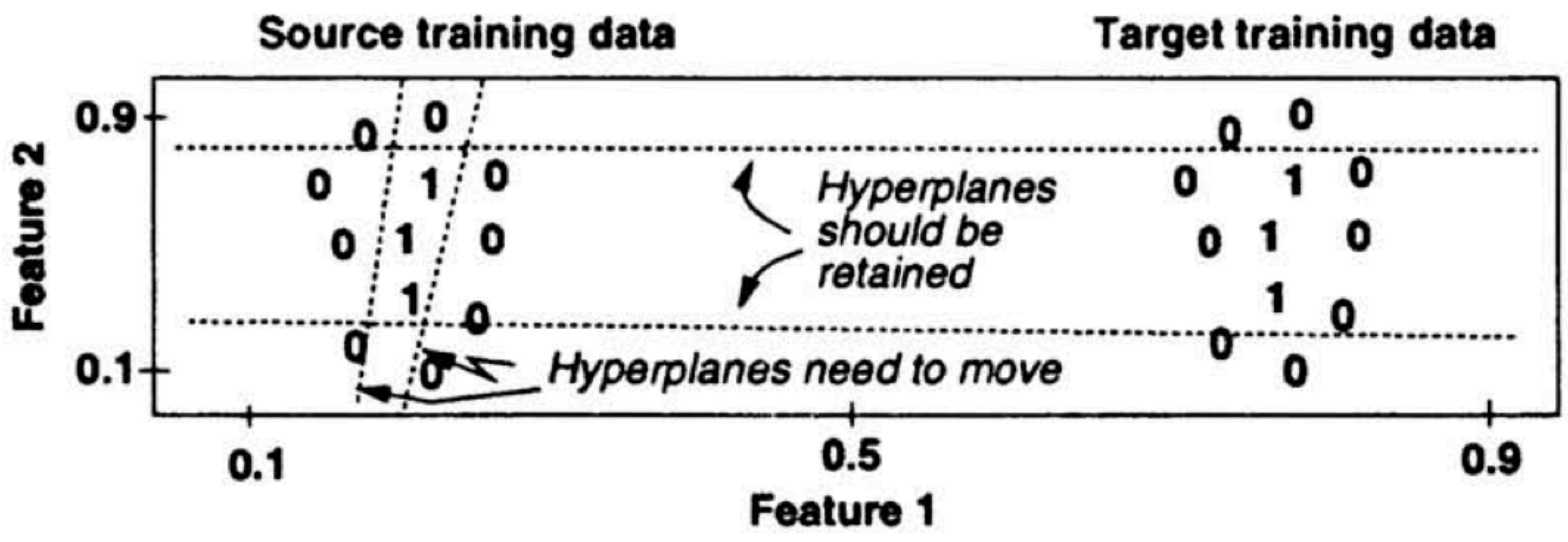
1. What to transfer: some knowledge is domain or task specific or may be more general/ transferable
2. When to transfer: when does transfer help or when does it even hurt?
3. How to transfer: algorithms to actually include, transfer/combine knowledge

(Some) **central objectives**

1. Improved loss/more accurate function in direct comparison to learning just on the target
2. Accelerate learning
3. Reduce data dependence (of target)

Examples of transfer learning approaches

Transductive transfer



Early approaches transfer by identifying the amount that a specific hyperplane helps to separate the data into different classes (& then reweighting/reinitializing).

"Discriminability-Based Transfer between Neural Networks", L. Y. Pratt, NeurIPS 1992

Transductive transfer

A domain adaptation example through feature transformation

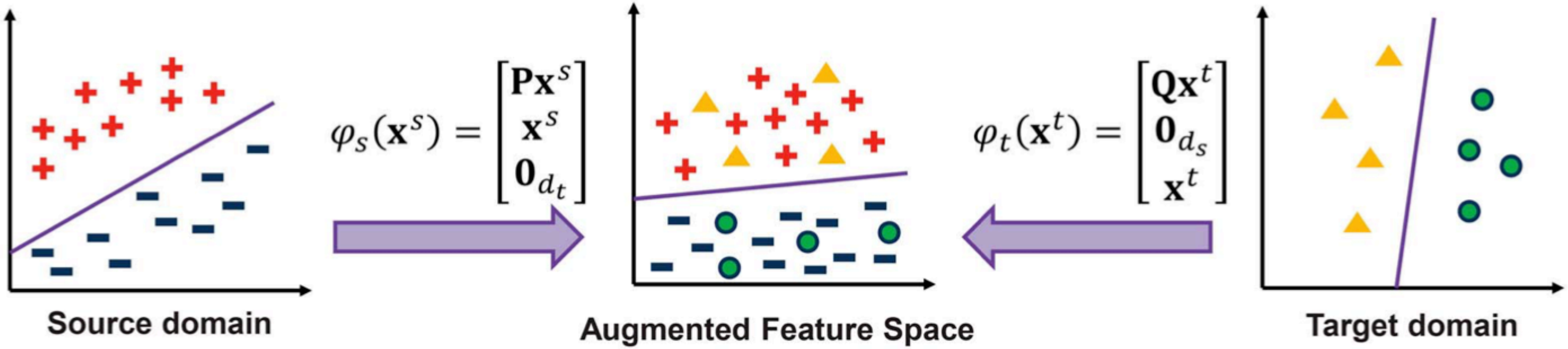
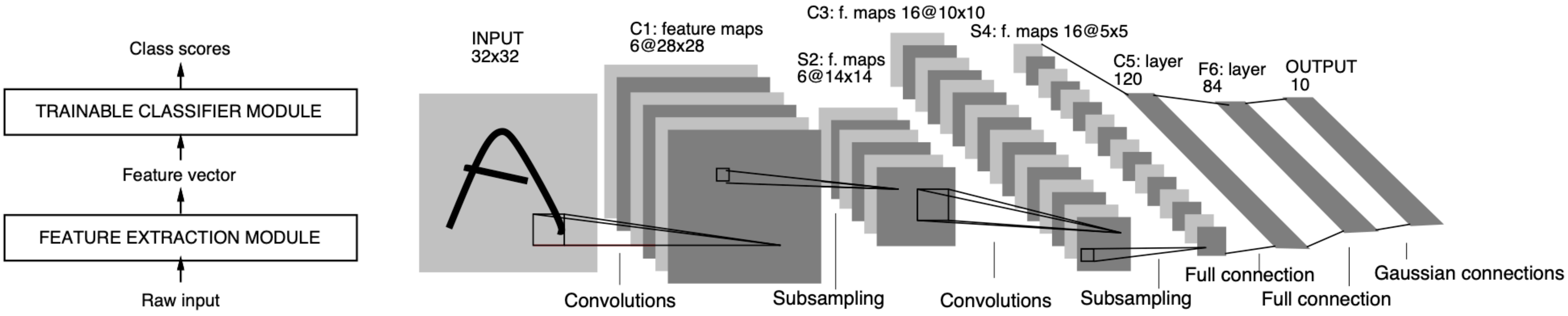


Fig. 1. Samples from different domains are represented by different features, where red crosses, blue strips, orange triangles and green circles denote source positive samples, source negative samples, target positive samples and target negative samples, respectively. By using two projection matrices **P** and **Q**, we transform the heterogenous samples from two domains into an augmented feature space.

"Learning with augmented Features for Supervised and Semi-Supervised Heterogeneous Domain Adaptation", Wen Li et al, TPAMI 2014

A small interlude/recap: convolutional neural networks

A small recap: convolutional NN



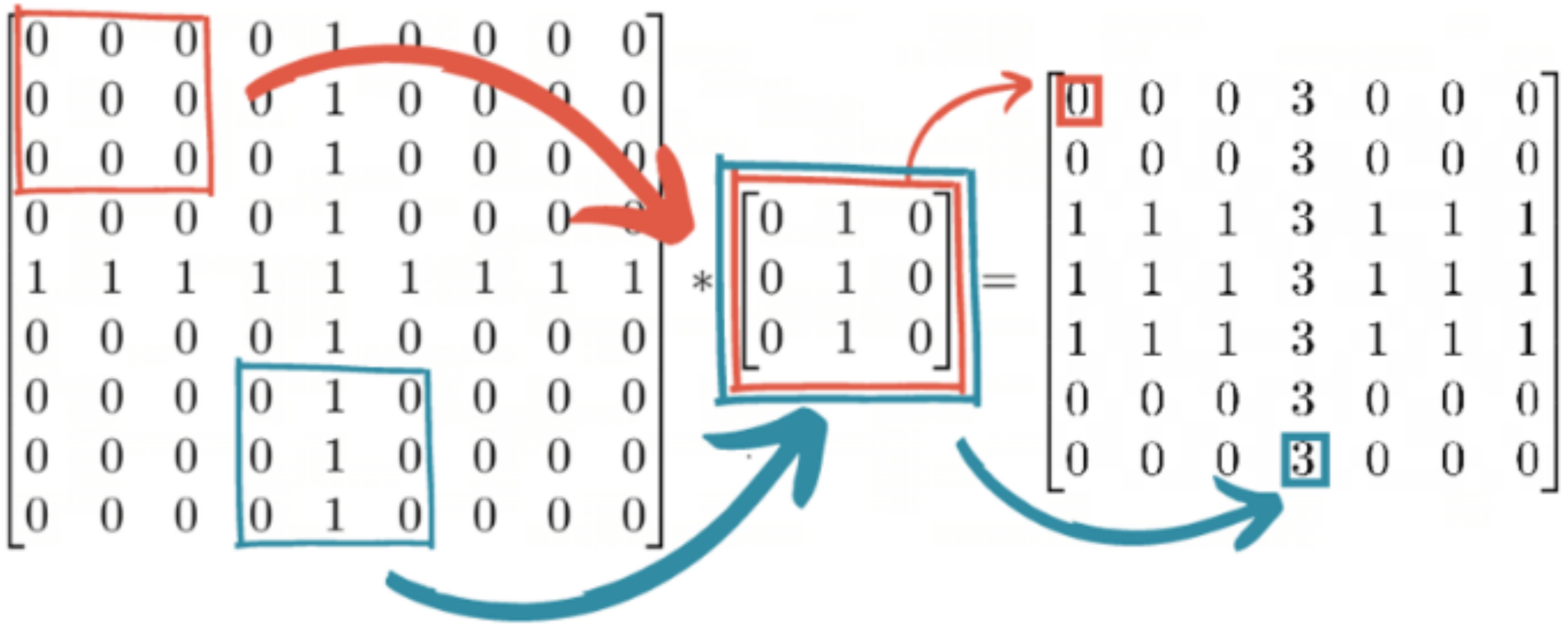
- **Convolutions:** multiple learnable patterns, “weight-sharing” - sliding window
- **Pooling**: not learned dimensionality reduction, also e.g. local invariance
- Modern advances like dropout, batch-norm etc. of which some are learnable
- If you don’t know how learning/training works, don’t worry, we’ll visit this next week

“Gradient-Based Learning Applied to Document Recognition”, LeCun et al, Proceedings of the IEEE, 1998

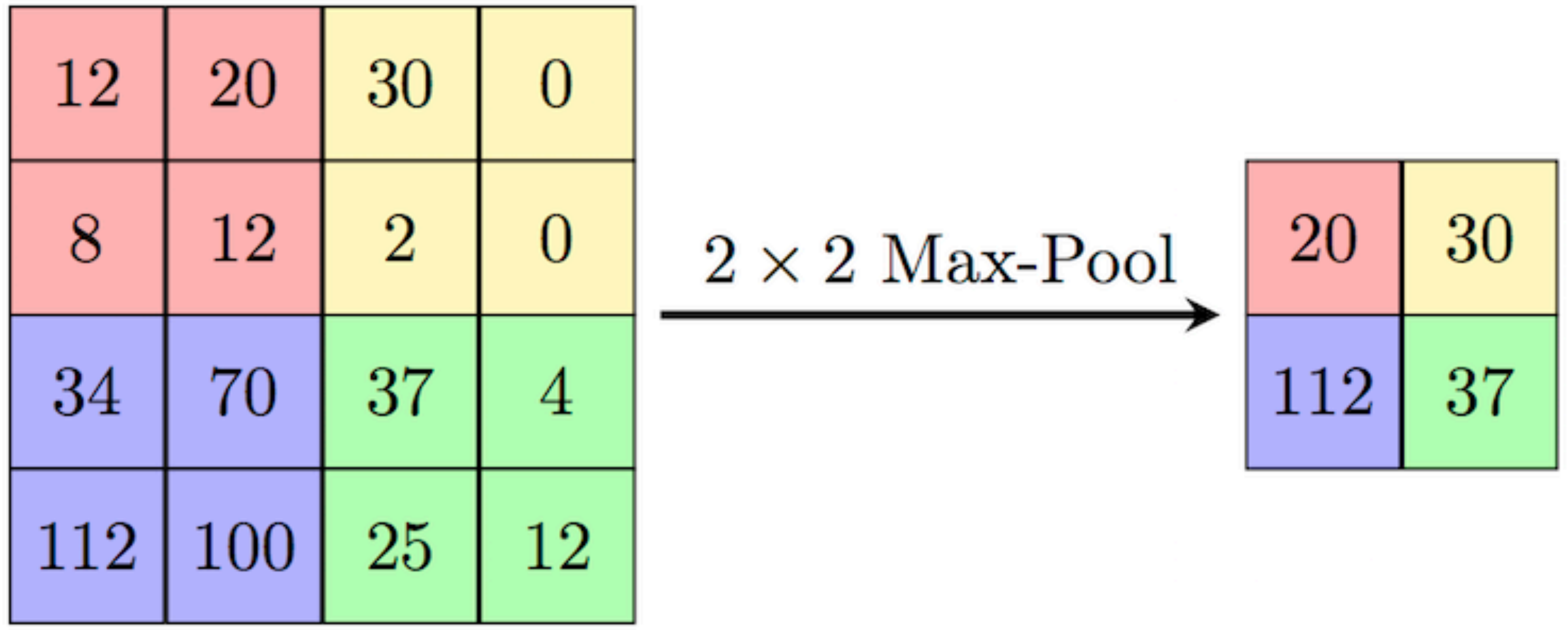
A small recap: convolutional NN

Convolutions: multiple learnable patterns, "weight-sharing" - sliding window

Pooling: not learned dimensionality reduction, also e.g. local invariance



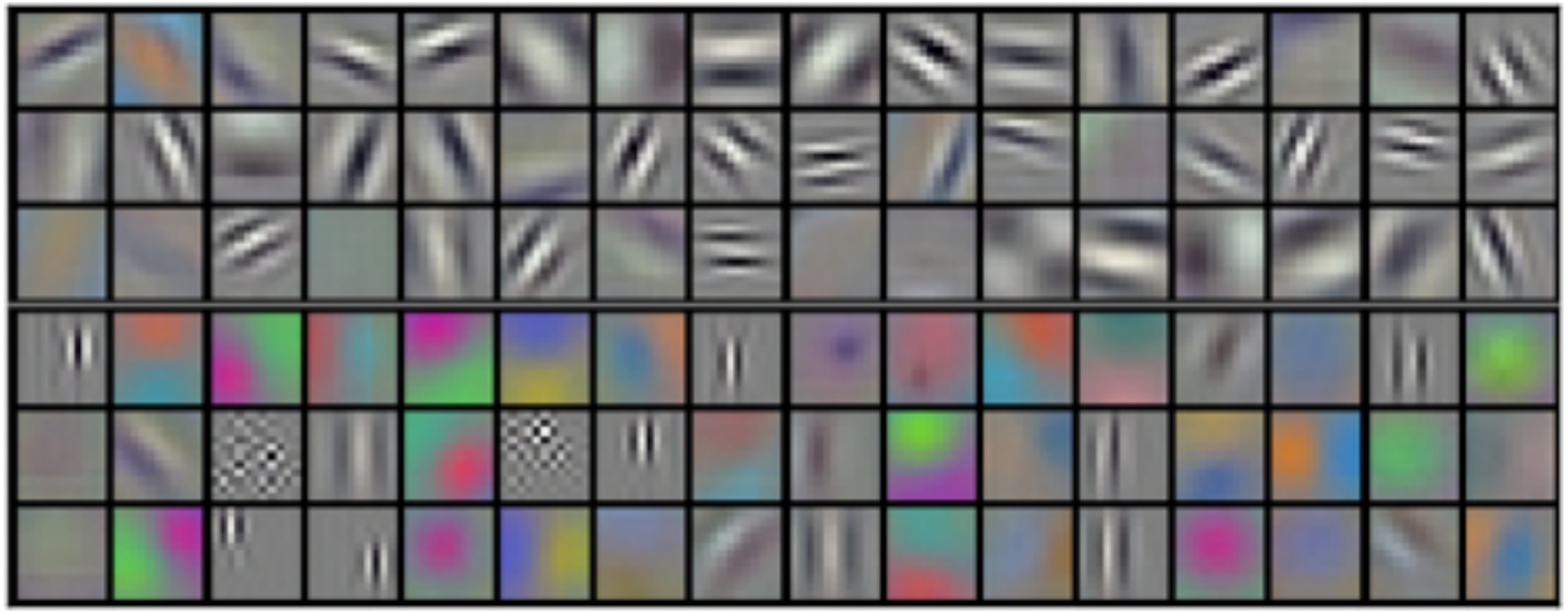
https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network



https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

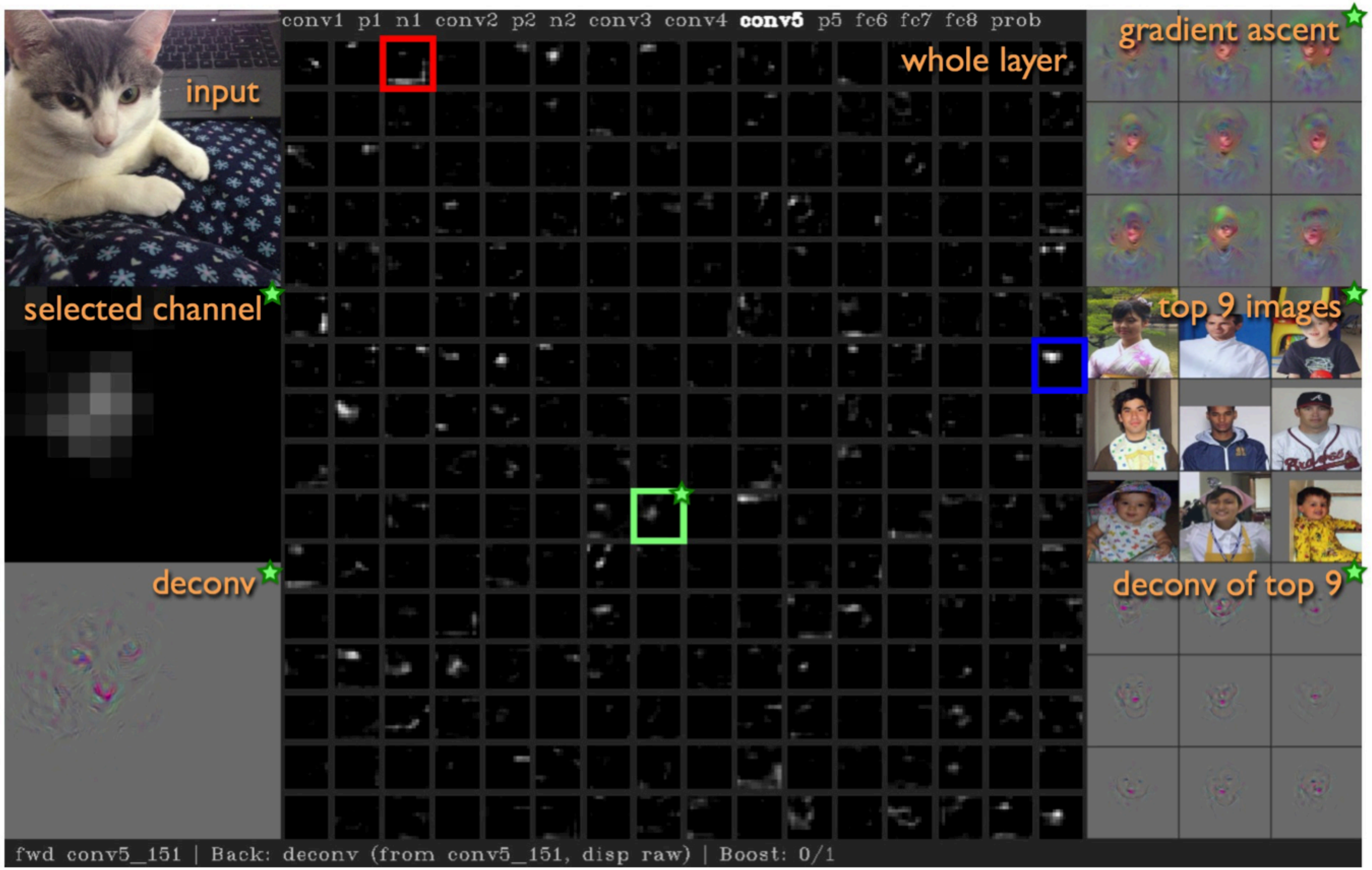
A small recap: convolutional NN

Key hypothesis: early layers learn generic patterns, deeper layers become increasingly more specific



"ImageNet Classification with Deep Convolutional Neural Networks", Krizhevsky et al, NeurIPS 2012

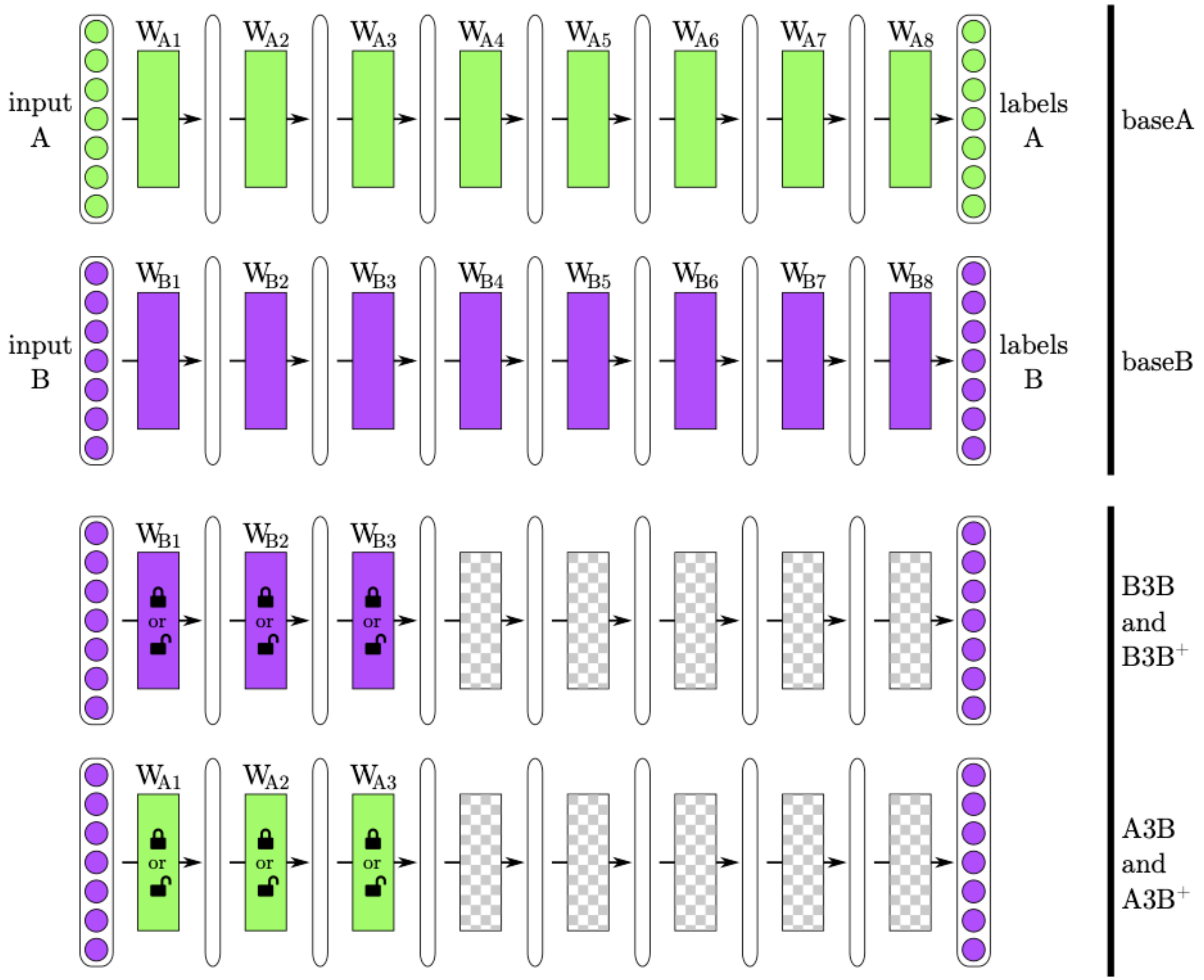
A small recap: convolutional NN



“Understanding Neural Networks Through Deep Visualization”, Yosinski et al, ICML Deep Learning Workshop, 2015

Transfer learning in deep learning

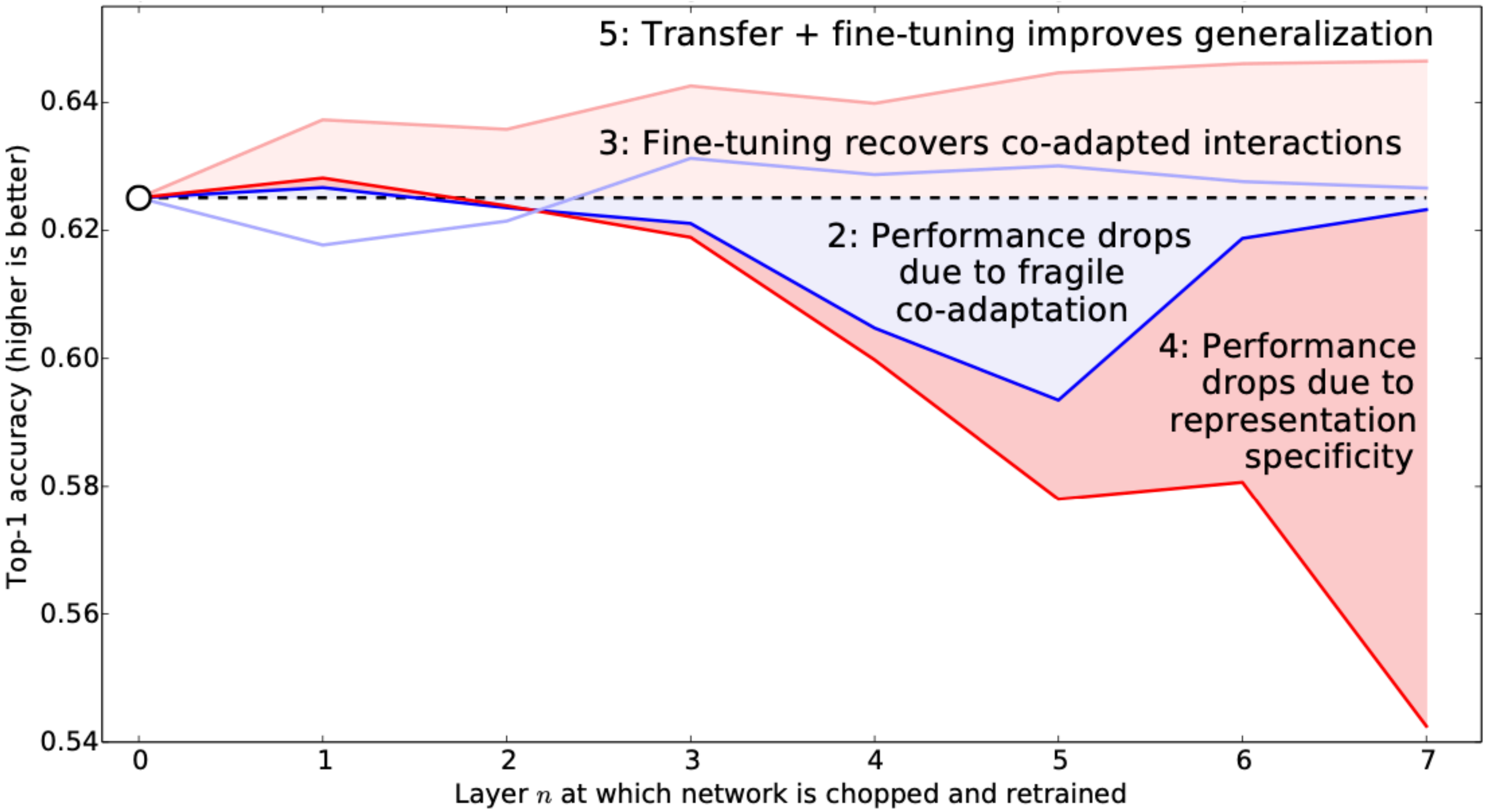
(Inductive) ImageNet transfer



- Split Imagenet into 2 sets of 500 classes: A and B
- “Lock” different sets of layers/representations & randomly initialize upper remaining layers
- Alternatively: continue training/fine-tuning transferred layers

“How transferable are features in deep neural networks”, Yosinski et al, NeurIPS 2014

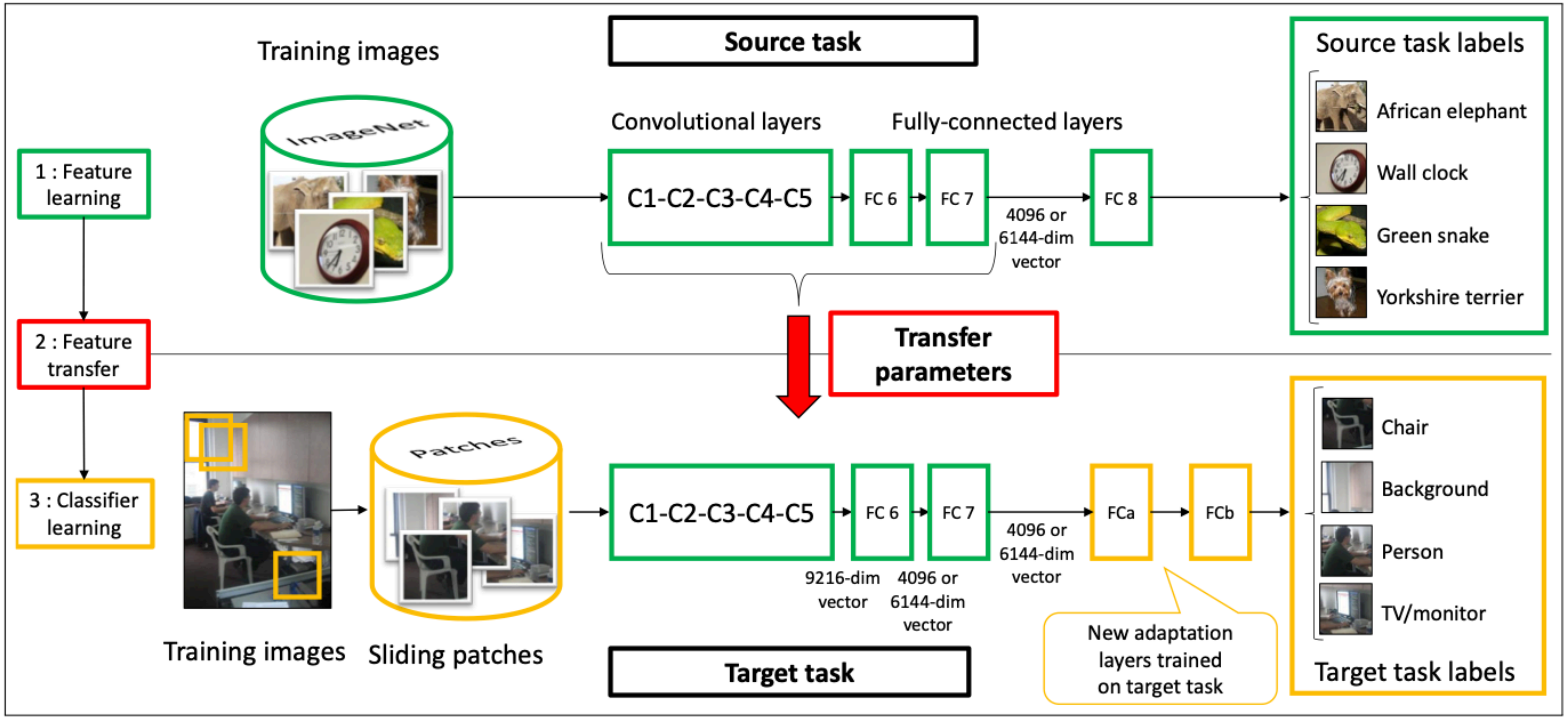
(Inductive) ImageNet transfer



2. B-B: copied from B and frozen + random rest trained on B
3. B-B+: copied features are allowed to adapt/fine-tune
4. A-B: transfer from A to B with frozen layers
5. A-B+: transferring + fine-tuning from A to B

“How transferable are features in deep neural networks”, Yosinski et al, NeurIPS 2014

(Inductive) ImageNet transfer



"Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks", Oquab et al, CVPR 2014

(Inductive) ImageNet transfer

Pre-training on ImageNet (e.g. 59 bird species and 120 dog breeds)
for the task on Pascal VOC 2012 (bird and dog class)

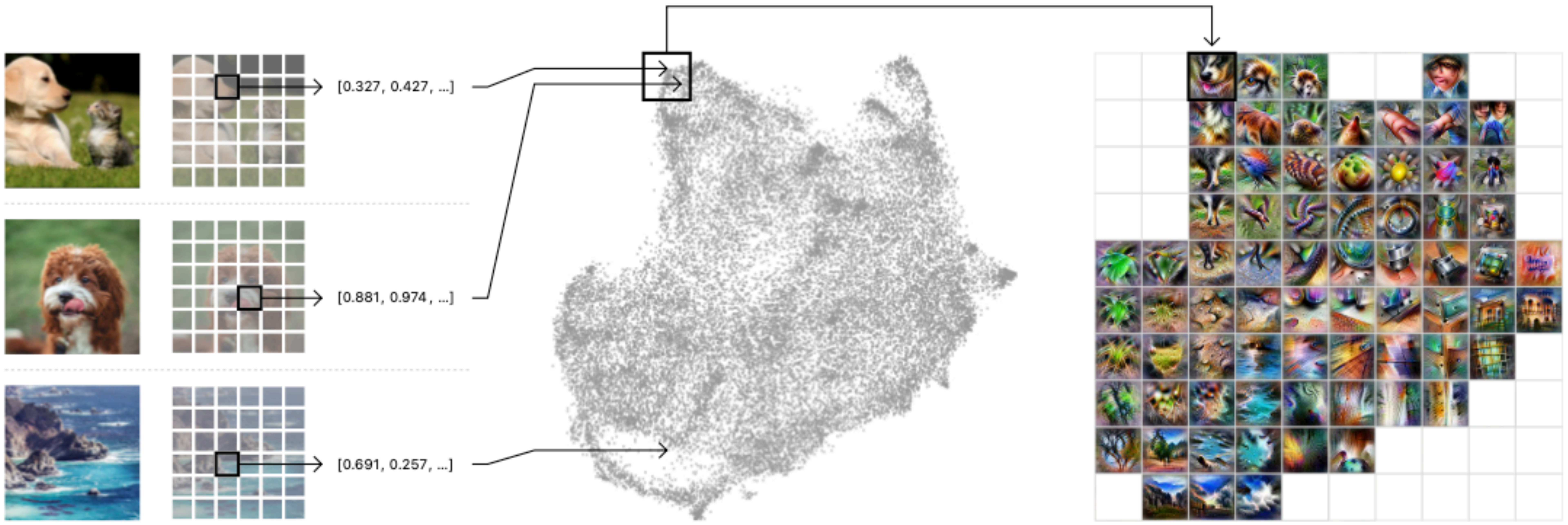
	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	table	dog
NUS-PSL [51]	**97.3**	**84.2**	80.8	**85.3**	**60.8**	**89.9**	**86.8**	89.3	**75.4**	77.8	**75.1**	83.0
NO PRETRAIN	85.2	75.0	69.4	66.2	48.8	82.1	79.5	79.8	62.4	61.9	49.8	75.9
PRE-1000C	93.5	78.4	*87.7*	80.9	57.3	85.0	81.6	*89.4*	66.9	73.8	62.0	*89.5*
PRE-1000R	93.2	77.9	83.8	80.0	55.8	82.7	79.0	84.3	66.2	71.7	59.5	83.4
PRE-1512	94.6	82.9	**88.2**	84.1	60.3	89.0	84.4	**90.7**	72.1	**86.8**	69.0	**92.1**

ImageNet Pascal VOC

"Learning and Transferring Mid-Level Image Representations using
Convolutional Neural Networks", Oquab et al, CVPR 2014

The role of embeddings: few-shot and one-shot transfer

The role of embeddings



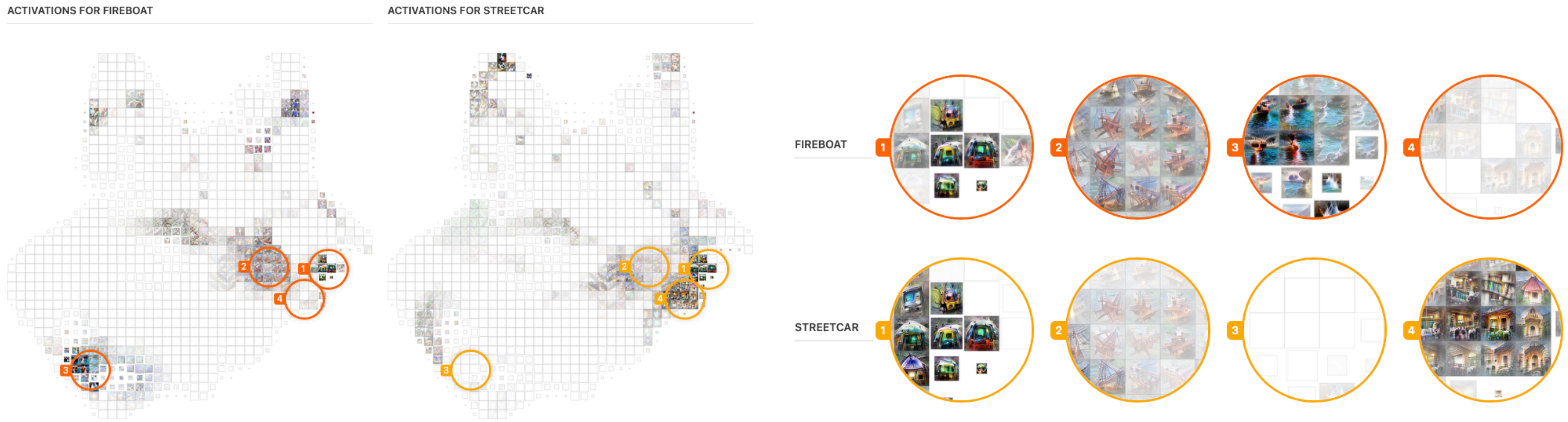
A randomized set of one million images is fed through the network, collecting one random spatial activation per image.

The activations are fed through UMAP to reduce them to two dimensions. They are then plotted, with similar activations placed near each other.

We then draw a grid and average the activations that fall within a cell and run feature inversion on the averaged activation. We also optionally size the grid cells according to the density of the number of activations that are averaged within.

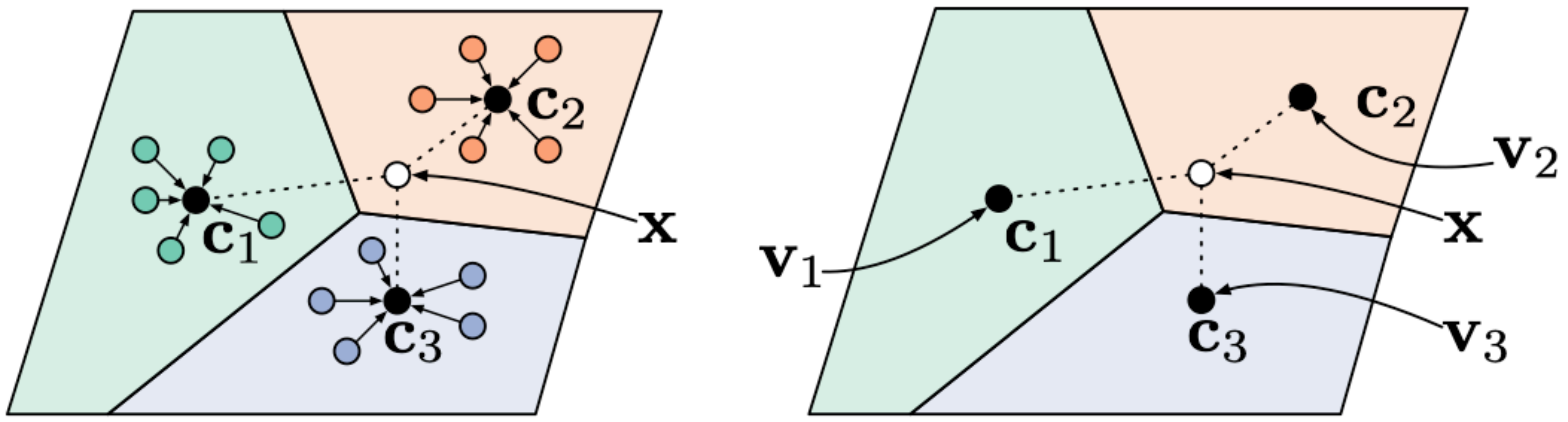
“Activation Atlas”, Carter et al, Distill 2019

The role of embeddings



"Activation Atlas", Carter et al, Distill 2019

Few-shot learning



(a) Few-shot (b) Zero-shot

Figure 1: Prototypical networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes $\mathbf{c}_k$ are produced by embedding class meta-data $\mathbf{v}_k$. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_\phi(y = k|\mathbf{x}) \propto \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))$.

Compute prototype c as the mean vector of each class with parametrized embedding function of a support set of labelled examples

Given a distance function d, classify according to softmax over distances to the prototypes in embedding space

“Prototypical Networks for Few-shot Learning”, Snell et al, NeurIPS 2017

See also “Object Classification from a Single Example Utilizing Class relevance Metrics”, M. Fink, NeurIPS 2004 & “One-shot Learning of Object Categories”, Fei-Fei et al, TPAMI 2006

One-shot learning

"*We say that a set of classes is* $\gamma > 0$ *separated with respect to a distance function d if for any pair of examples belonging to the same class* $\{(x_1, c), (x_1', c)\}$*, the distance* $d(x_1, x_1')$ *is smaller than the distance between any pair of examples from different classes* $\{(x_2, e), (x_2', g)\}$ *by at least* γ: $d(x_1, x_1') \leq d(x_2, x_2') - \gamma$."

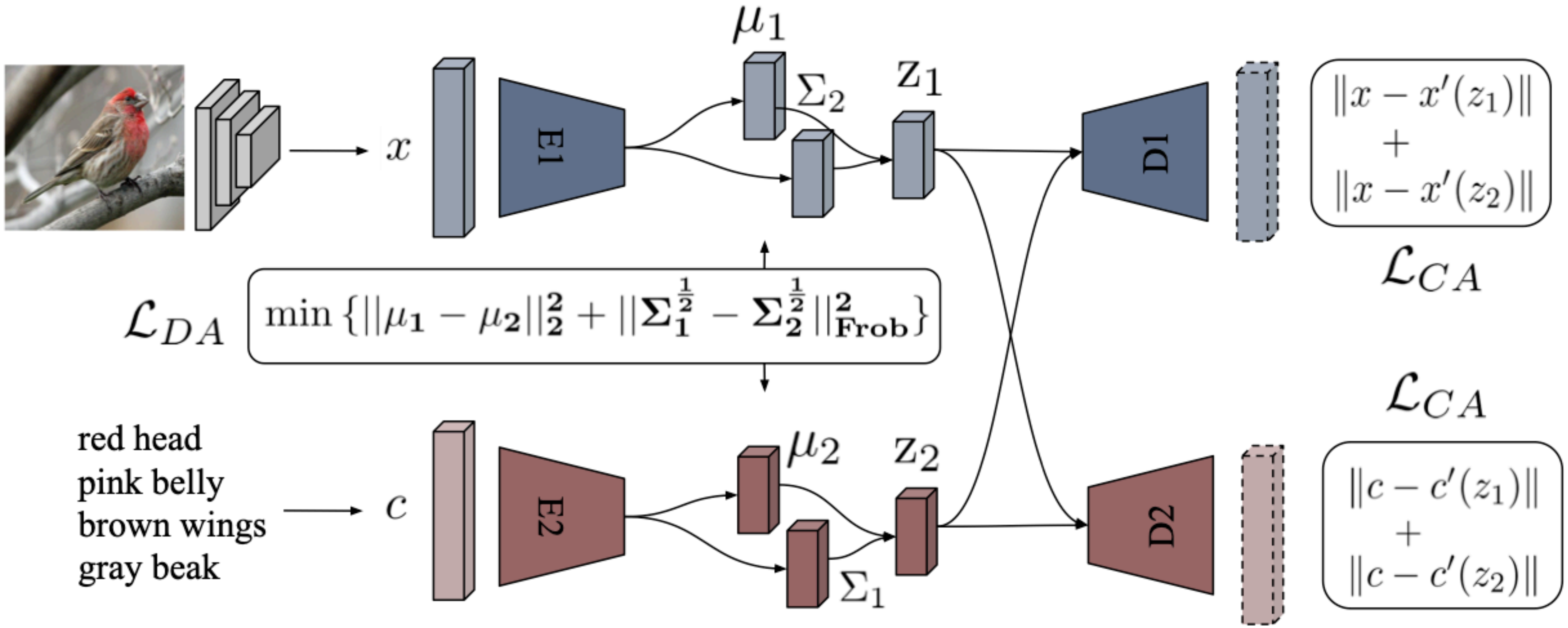
1. Learn from extra sample a distance function d that achieves gamma separation
2. Learn a nearest neighbor classifier, where the classifier employs d

"Object Classification from a Single Example Utilizing Class relevance Metrics", M. Fink, NeurIPS 2004

See also "One-shot Learning of Object Categories", Fei-Fei et al, TPAMI 2006

Few-shot learning

Common approach: measures of maximum mean discrepancy or Wasserstein distance

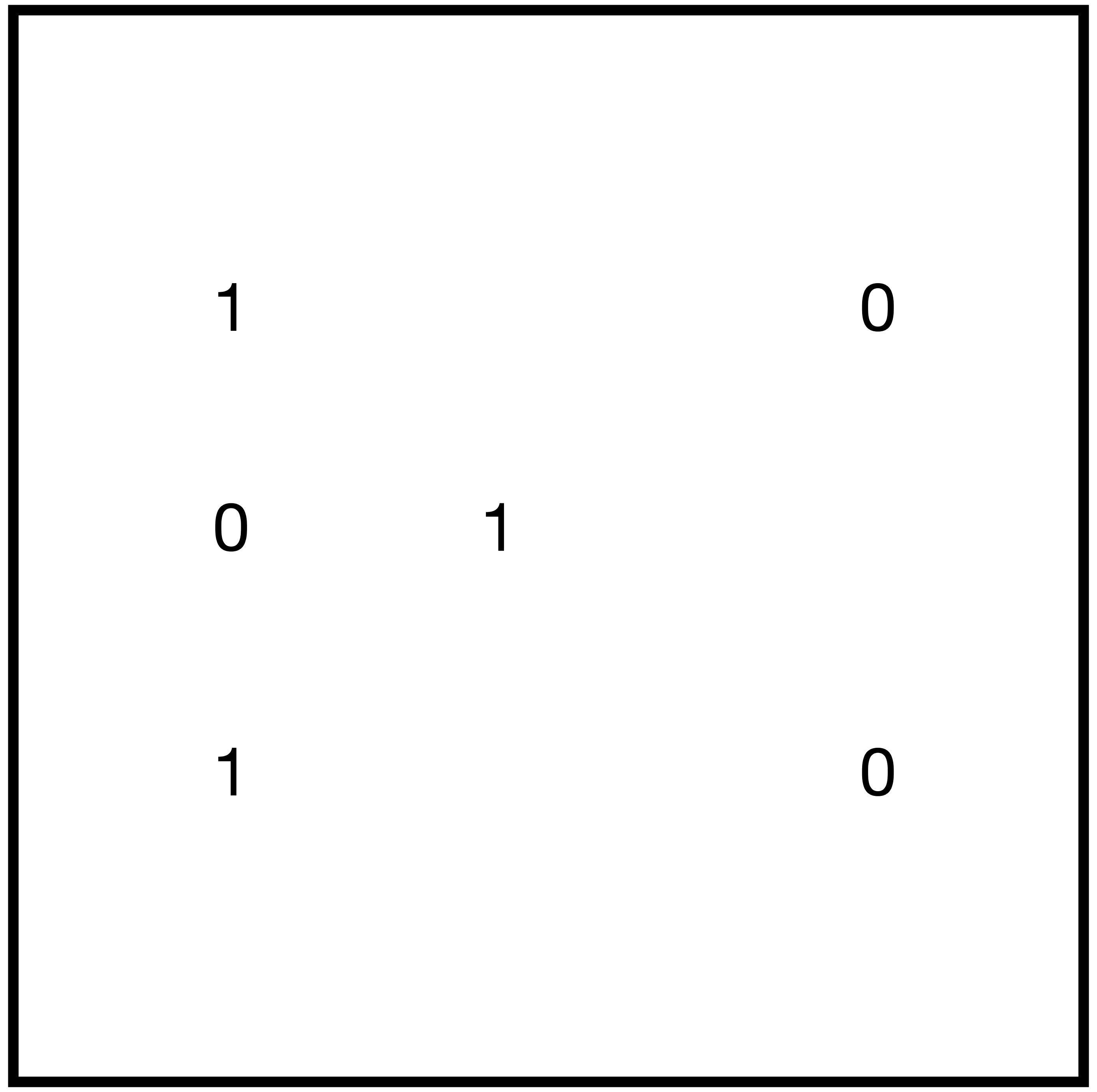


“Generalized Zero- and Few-Shot Learning via Aligned Variational Autoencoders”, Schoenfeld et al, CVPR 2019

Why is transfer challenging?

Transfer challenges

How would you separate this data with a set of hyperplanes? (Try 3)



Transfer challenges

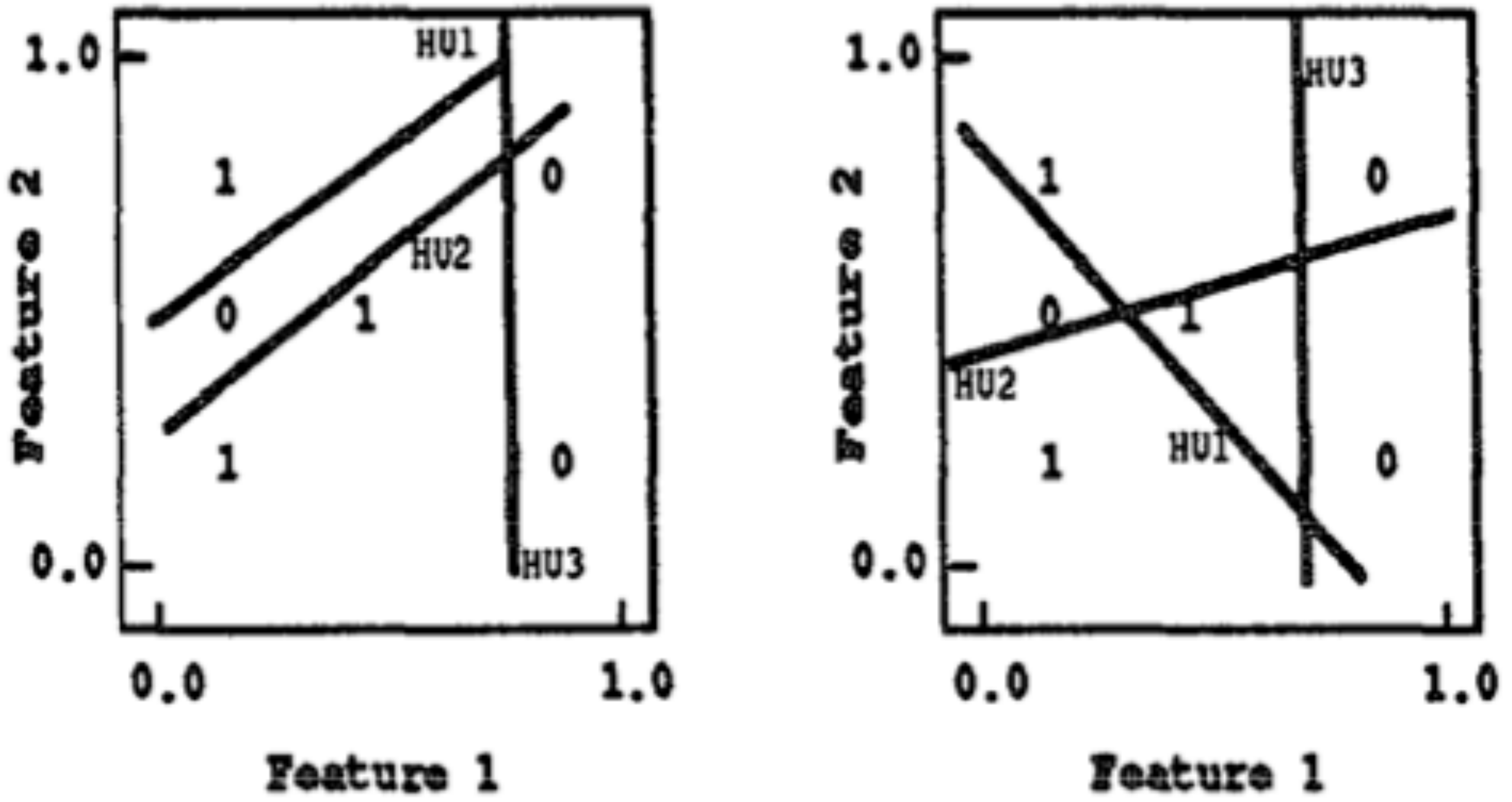
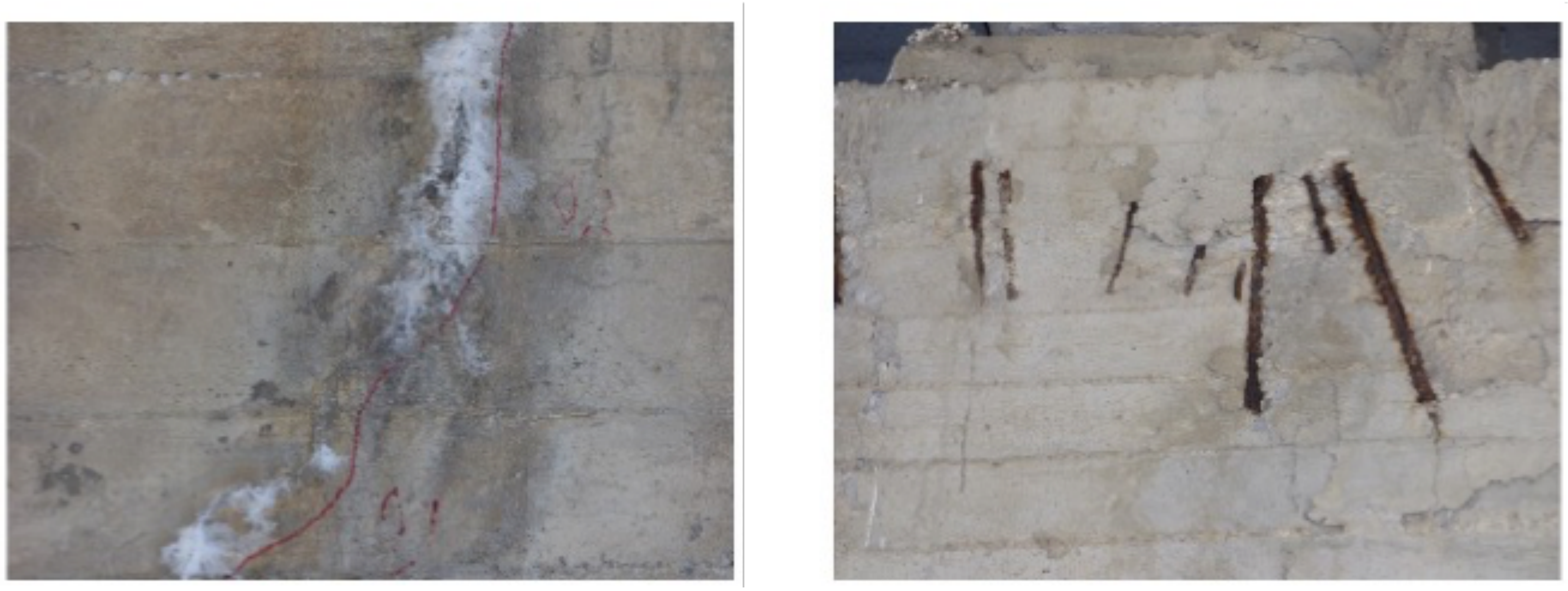


Figure 2: Two examples of hyperplane sets that separate training data in a small network.

“Direct Transfer of Learned Information Among Neural Networks” , L. Y. Pratt et al, AAAI 1991

Not intuitive if transfer works



“Meta-learning Convolutional Neural Architectures for Multi-target Concrete Defect Classification with the Concrete Defect Bridge Image Dataset”, Mundt et al, CVPR 2019

<—Training from scratch:

- Alexnet: 66.98 %
- VGG-A: 70.45%
- VGG-D: 70.61%

Transfer learning

Architecture	Source	Accuracy [%]
Alexnet	ImageNet	62.87
VGG-A	ImageNet	66.35
VGG-D	ImageNet	65.56
Densenet-121	ImageNet	57.66
Alexnet	MINC	66.50
VGG-D	MINC	67.14

“Material Recognition in the Wild with the Materials in Context Database, CVPR 2015”

Is selective transfer a solution?

Alternatively to selecting entire layers, freezing the weights or letting them partially adapt, we could also try to select and inject only the features that are "representative" for the new task

- For instance: pick only features that have large activations

TABLE III: Performance (accuracy) comparison for different tasks. *M*: Material features learned using MINC. *O*: Object features learned using ILSVRC2012. *MO*: Concatenated material and object features ($\mathbf{x}_c \in \mathcal{F}$). *SMO*: Features integrated using the proposed method ($\mathbf{x}_c \in \mathcal{S}$).

Task	*M* (%)	*O* (%)	*MO* (%)	*SMO* (%)
FMD	80.4 ± 1.9	79.6 ± 2.1	79.1 ± 2.5	$\mathbf{82.3 \pm 1.7}$
FMD-2	82.5 ± 2.0	82.9 ± 1.6	83.9 ± 1.8	$\mathbf{84.0 \pm 1.8}$
EFMD	88.7 ± 0.2	88.8 ± 0.3	$\mathbf{89.7 \pm 0.13}$	$\mathbf{89.7 \pm 0.16}$
MINC-val	82.45 [22]	68.17	83.48	**83.93**
MINC-test	82.19 [22]	68.04	83.12	**83.60**

"Integrating Deep Features for Material Recognition", Zhang et al, ICPR 2016

Representation Bias

Representations are biased in ways that we don't anticipate: **texture bias**

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

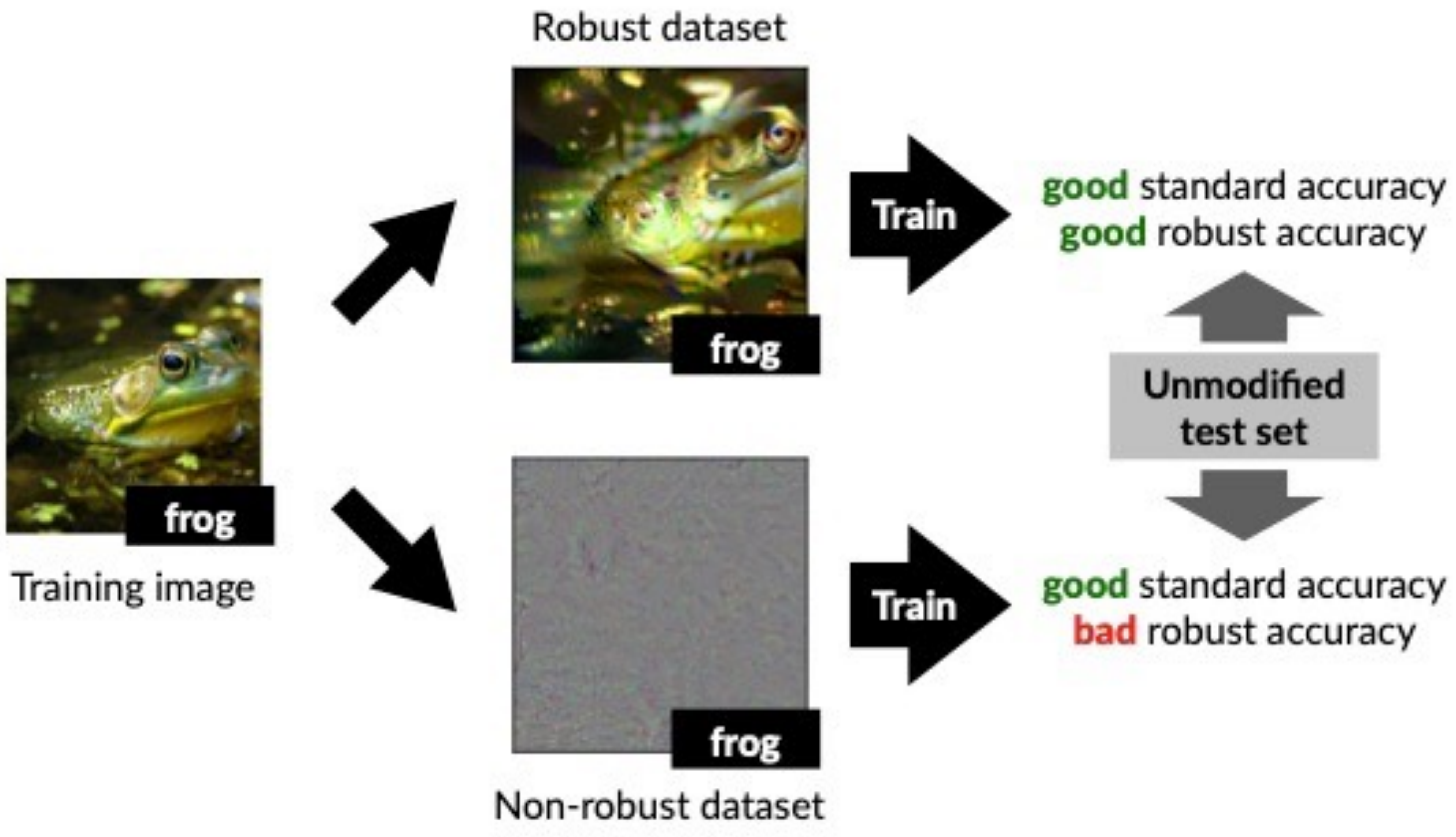
(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

"ImageNet-trained CNNS are biased towards texture", Geirhos et al, ICLR 2019

Adversarial features

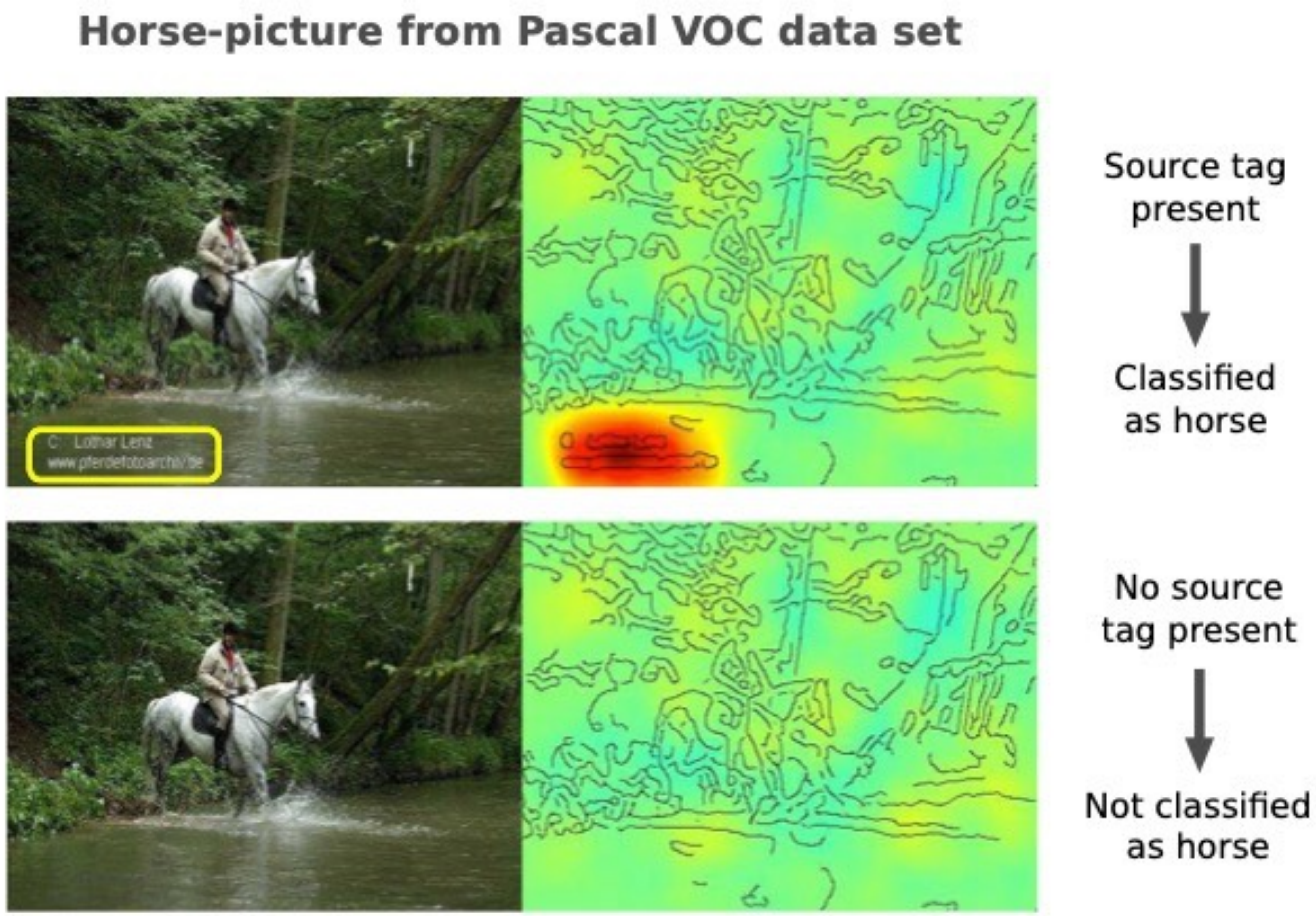
Representations are biased in ways that we don't anticipate: **adversarial** features



"Adversarial Examples are not Bugs, they are Features", Ilyas et al, NeurIPS 2019

Clever Hans predictors

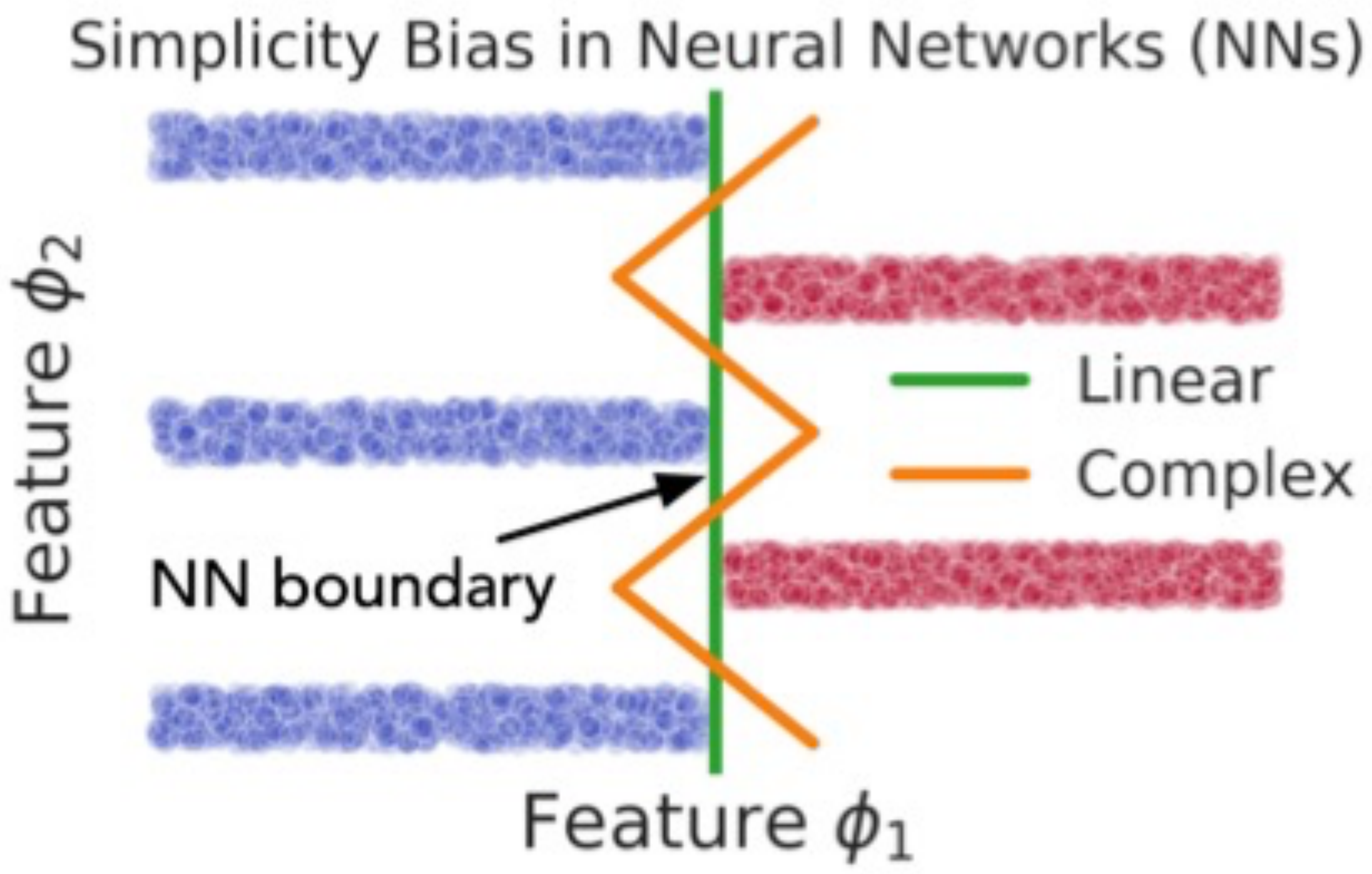
Representations are often biased in ways that we don't anticipate: **confounders**



"Unmasking Clever Hans Predictors", Lapuschkin et al, Nature Communications 2019

Simplicity bias

Representations are often biased in ways that we don't anticipate: pitfalls of **simplicity**



"The Pitfalls of Simplicity Bias in Neural Networks", Shah et al, NeurIPS 2020

Can you think of other ways to transfer knowledge?

Learning from "hints"

"A hint is any piece of information about the function f. As a matter of fact, an input-output example is a special case of a hint. A hint may take the form of a global constraint on f, such as a symmetry property or an invariance."

Abu-Mostafa, "Learning from Hints in Neural Networks"
Journal of Complexity 6, 1990

Symmetries

We could directly include in- or equivariance into our representations (e.g. rotation)

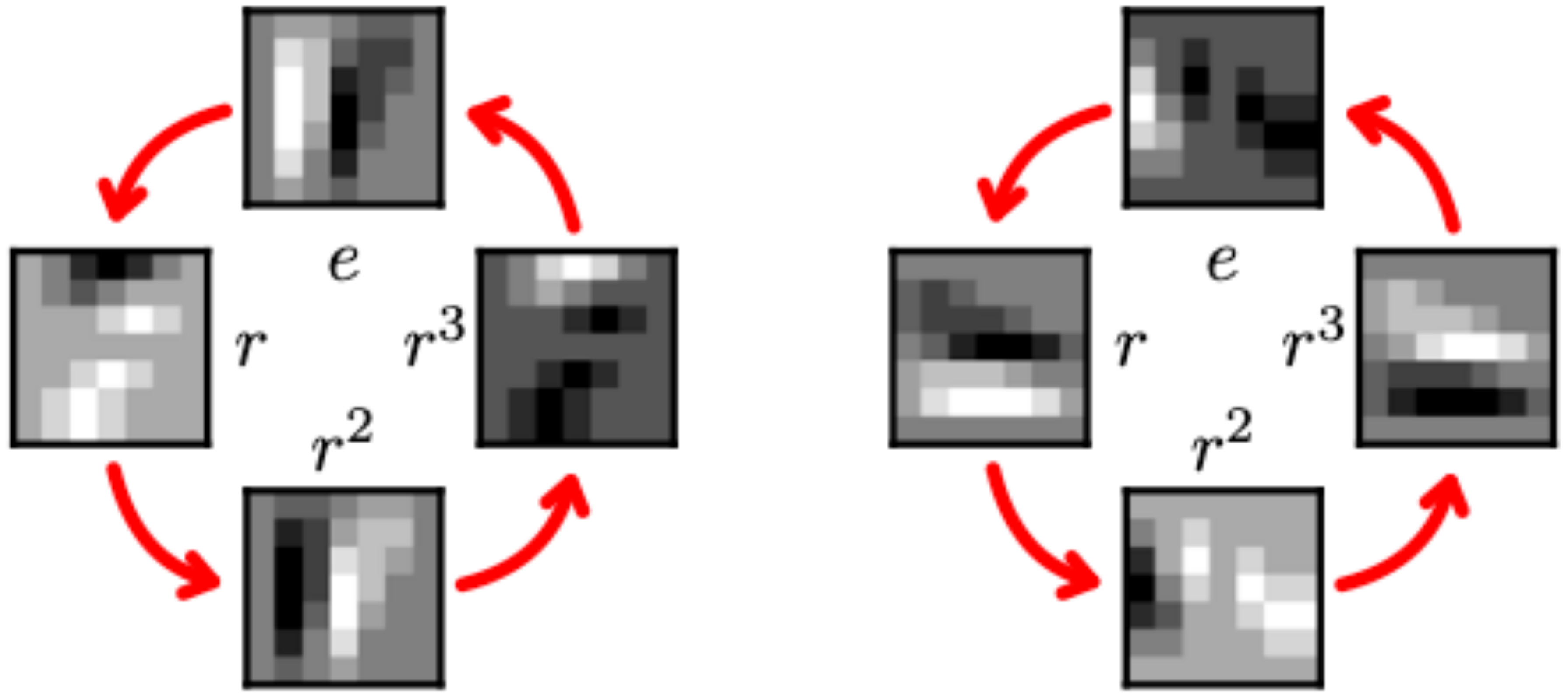
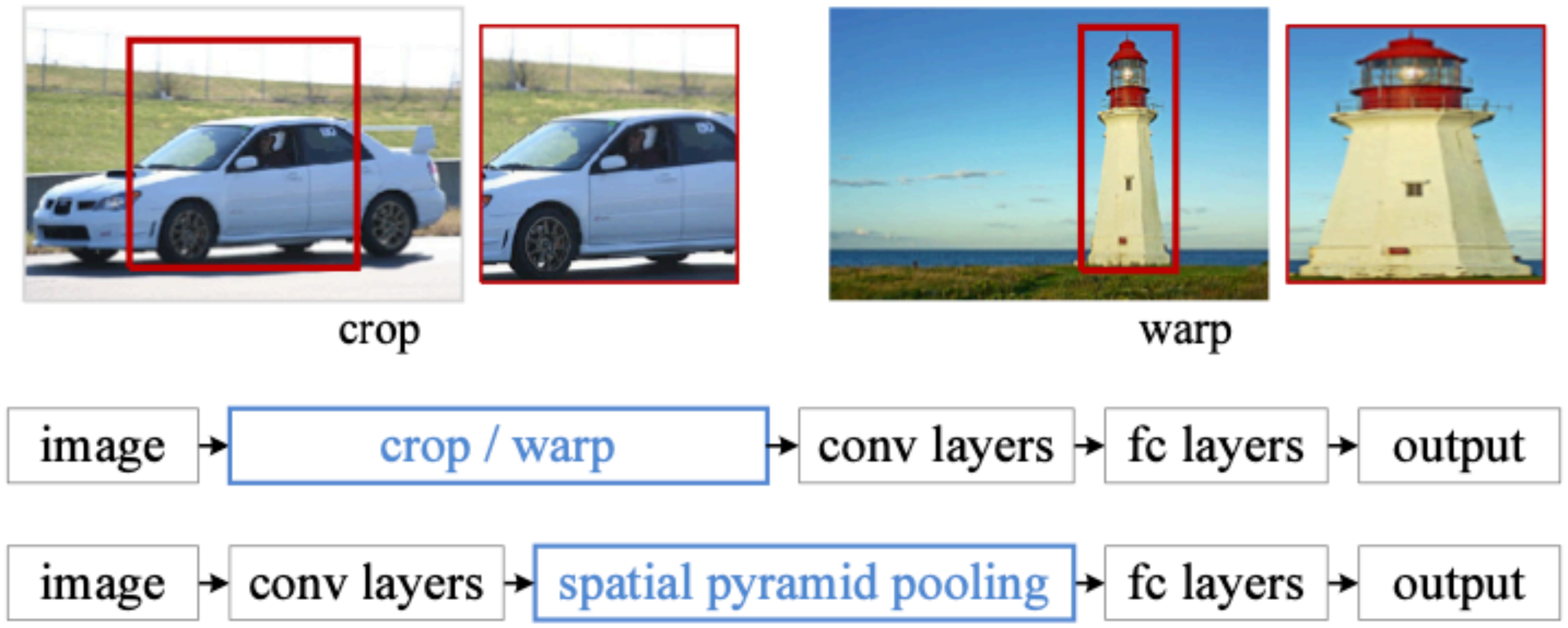


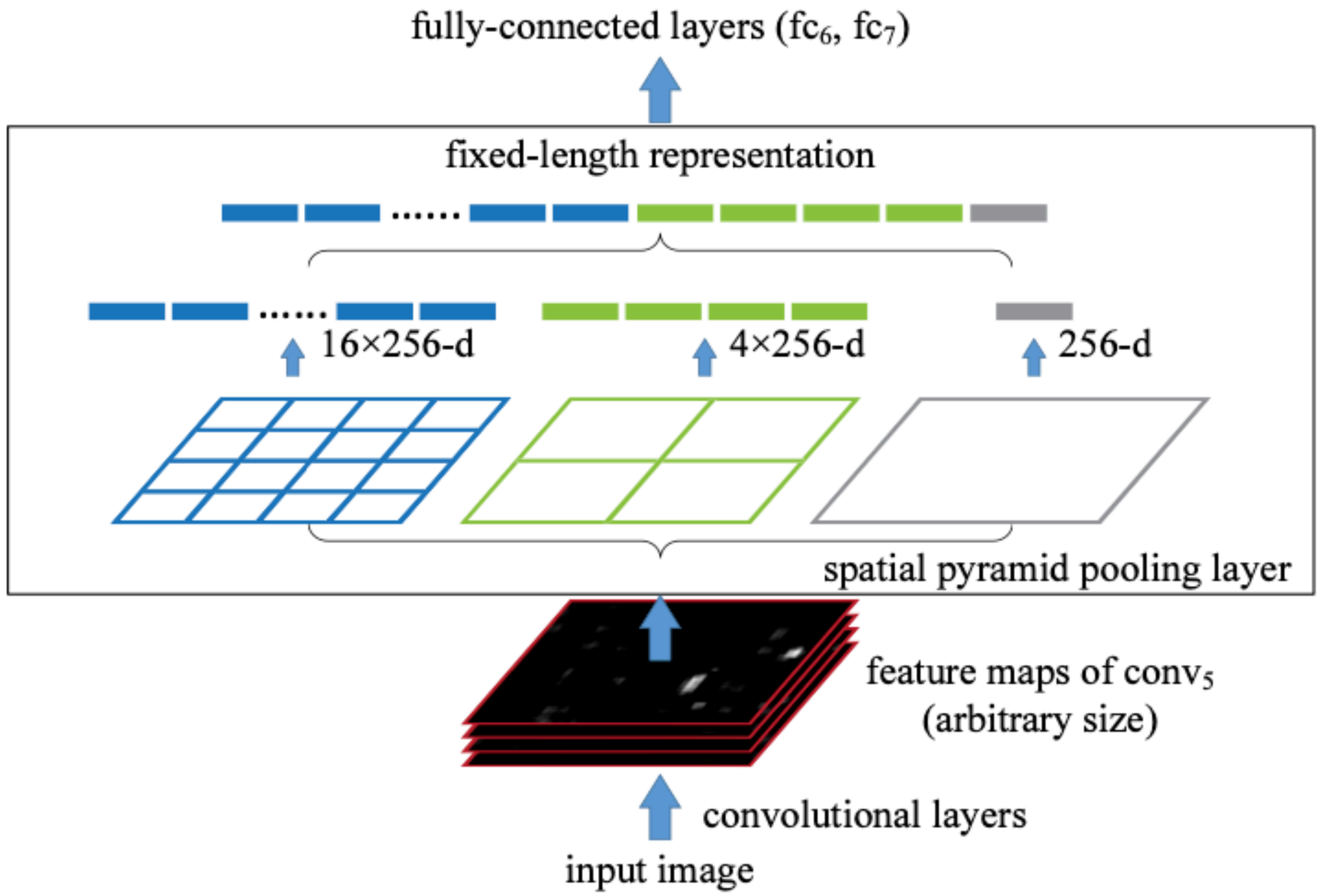
Figure 1. A p4 feature map and its rotation by r.

“Group Equivariant Convolutional Networks”, Cohen & Welling, ICML 2016

Making use of invariance

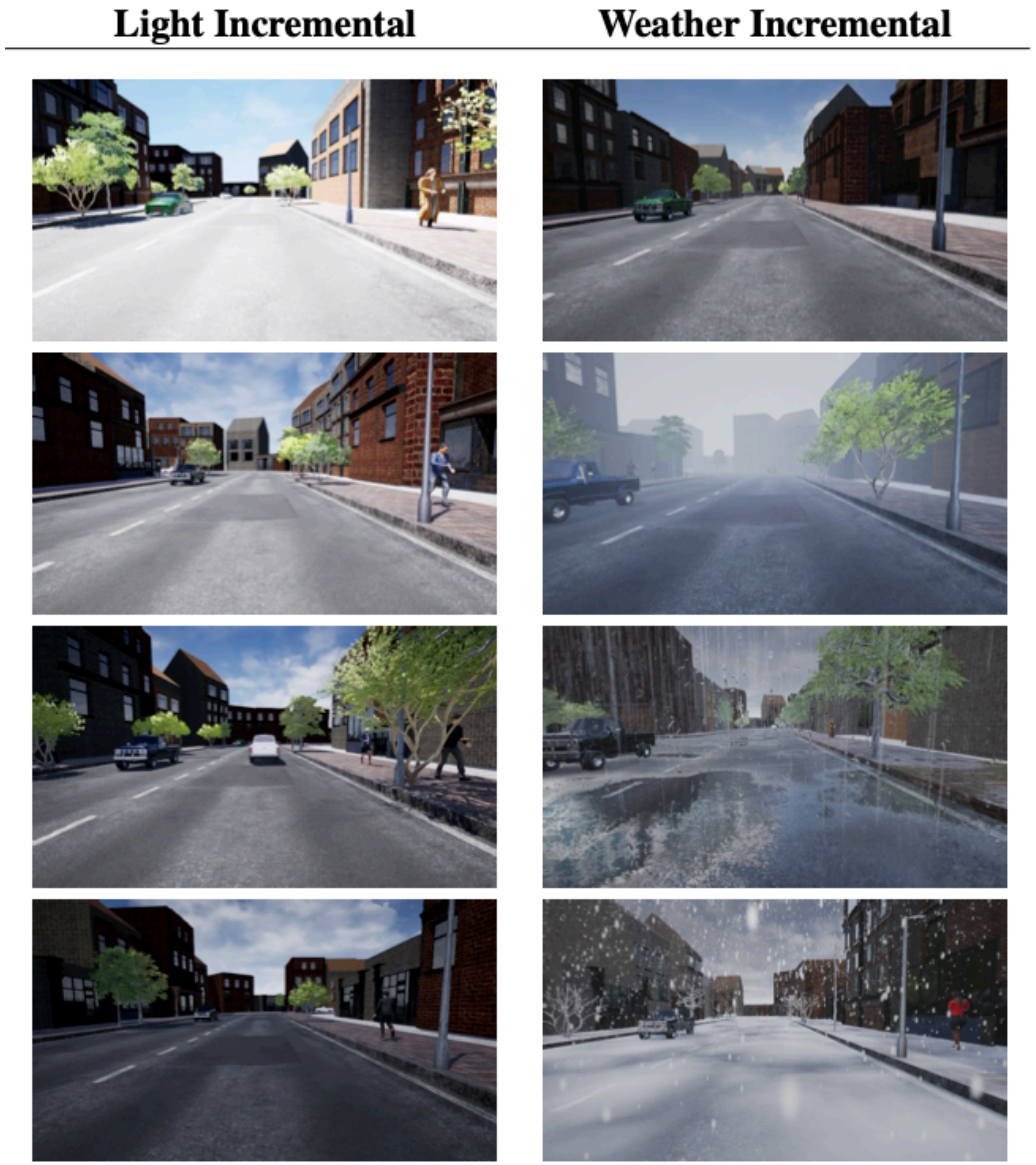
We could also incorporate a degree of scale invariance, or even try to learn it





"Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", K. He et al, TPAMI 2015

Making use of invariance



We could pre-process & account for illumination changes

Table 1: Incremental lighting experiment under consideration of a photometric color invariant or local binary patterns (LBP).

Illumination Intensity [Lux]	Accuracy [%] Naive	Naive + photometric color invariant	Naive + LBP
76.8	99.20 $\pm^{0.1}_{0.1}$	98.66 $\pm^{0.15}_{0.19}$	99.18 $\pm^{0.06}_{0.05}$
19.2	97.11 $\pm^{1.20}_{1.46}$	98.61 $\pm^{0.47}_{0.98}$	99.27 $\pm^{0.12}_{0.09}$
9.6	93.55 $\pm^{2.58}_{2.7}$	98.61 $\pm^{0.21}_{0.36}$	99.26 $\pm^{0.07}_{0.05}$
2.4	91.55 $\pm^{1.00}_{0.14}$	97.56 $\pm^{0.76}_{0.76}$	99.42 $\pm^{0.05}_{0.03}$
1.2	90.89 $\pm^{1.61}_{2.39}$	95.28 $\pm^{1.32}_{2.07}$	99.40 $\pm^{0.04}_{0.04}$

"A Procedural World Generation Framework for Systematic Evaluation of Continual Learning", Hess et al, NeurIPS 2021

Making use of invariance

We could assume that RGB color ratios are quasi invariant with white illumination:

$c_1 = \mathtt{arctan}(R/\mathtt{max}\{G, B\})$

(Gevers & Smeulders, “Color Based Object Recognition”, Pattern Recognition 32:3, 1999)

Making use of invariance

Alternatively: local binary patterns (He & Wang, Pattern Recognition 23:8, 1990) (simplified): mark all nearest neighbors for a pixel with 0 if greater and 1 otherwise, compute histogram over values to create features

Back to lifelong learning

Early definition: lifelong ML

Definition - **Lifelong Machine Learning** - Thrun 1996:

“The system has performed N tasks. When faced with the (N+1)th task, it uses the knowledge gained from the N tasks to help the (N+1)th task.”

- We have looked primarily at positive transfer today
- Let us now look at training & avoiding negative transfer (or forgetting)

“Is Learning The n-th Thing Any Easier Than Learning the First?” (NeurIPS 1996) & “Explanation based Neural Network Learning A Lifelong Learning Approach”, Springer US, 1996

Later definition: lifelong ML

Definition - **Lifelong Machine Learning** - Chen & Liu 2017:

"Lifelong Machine Learning is a continuous learning process. At any time point, the learner performed a sequence of N learning tasks, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$. These tasks can be of the same type or different types and from the same domain or different domains. When faced with the (N+1)th task $\mathcal{T}_{N+1}$ (which is called the new or current task) with its data D_{N+1}, the learner can leverage past knowledge in the knowledge base (KB) to help learn $\mathcal{T}_{N+1}$.
The objective of LML is usually to optimize the performance on the new task $\mathcal{T}_{N+1}$, but it can optimize any task by treating the rest of the tasks as previous tasks. KB maintains the knowledge learned and accumulated from learning the previous task. After the completion of learning $\mathcal{T}_{N+1}$, KB is updated with the knowledge (e.g. intermediate as well as the final results) gained from learning $\mathcal{T}_{N+1}$. The updating can involve inconsistency checking, reasoning, and meta-mining of additional higher-level knowledge."

"Lifelong Machine Learning", Chen & Liu, Morgan Claypool, 2017